MASTER'S THESIS

IRMA and network anonymity with Tot

Kreukniet, M.

Award date:
2021

IRMA AND NETWORK ANONYMITY WITH TOR

by

Markus Kreukniet

Student number:
Date of presentation: 30 April 2021

Open Universiteit
de beste! www.ou.nl

IRMA AND NETWORK ANONYMITY WITH TOR

by

Markus Kreukniet

Student number:		
Date of presentation:	30 April 2021	
Course code:	IM9906	
Graduation committee:		
Primary supervisor (chair):	Dr. Fabian van den Broek	Open University
Secondary supervisor:	Dr. Greg Alpár	Open University

Open University of the Netherlands, Faculty of Science
Master's Programme in Computer Science or Master's Programme in Software Engineering

CONTENTS

Abstract

IRMA is a privacy-friendly authentication platform since it provides privacy guarantees on the application layer. However, it has no such guarantees on the network layer. We can add privacy guarantees on the network layer by routing the IRMA context's network traffic over the Tor network. This routing adds network latency, and we want to measure how much the added latency is. A similar study measured whole sessions while the IRMA mobile app's network traffic was routed over the Tor network, using an external app, in two unrealistic scenarios. By measuring individual network messages of sessions, in more and realistic scenarios gives realistic network latencies and shows which network messages can be problematic. We implemented Tor in the IRMA mobile app and performed measurements in multiple situations, such as measurements over a WiFi network and a cellular network. The implementation itself gave insights usable for similar implementations, such as that the used libraries have limitations that are not entirely solvable with workarounds. The measurement results gave insight into the added network latency of using Tor in various situations. In realistic scenarios routing the app's network traffic over the Tor network can add an extra average network latency of 2,63 seconds. We recommend future research to find out if such added latency is acceptable. Further, this study compared implementing Tor in the IRMA mobile app and using Tor with an external Tor app.

Keywords: IRMA, IRMA mobile app, measurements, network latency, Tor, The Onion Router, Tor implementation

1. INTRODUCTION

IRMA is a privacy-friendly authentication platform, and the abbreviation stands for 'I Reveal My Attributes.' [Alpár et al., 2017]. Part of this platform is the IRMA mobile app, available on Android and iOS [Privacy by Design Foundation, 2020i]. On this app, users can store relevant information about themselves in the form of attributes. An attribute is the smallest part of the information a user can disclose. A possible attribute might be '16 years or older'. If a user wants to authenticate themselves to a server within the IRMA platform, the server can ask to disclose one or more user attributes. When a user chooses to disclose the asked attributes, the server authenticates the user by verifying the attributes. Users can disclose only relevant attributes with this authentication method without disclosing attributes that identify the user too much. For example, when someone wants to buy alcohol in the Netherlands. This person can prove that he/she is at least 18 years or older by only disclosing the attribute '18 years or older'. The authentication method contributes to the GDPR data minimization.

One of the reasons we can see IRMA as privacy-friendly is that users can authenticate themselves to a server by disclosing only relevant attributes. However, the authentication with attributes is only a privacy guarantee on the application layer, but not on the network layer [Alpár et al., 2017]. Essentially, the IP address is also disclosed in every IRMA transaction, which might function as a unique identifier or at least provide the means to link different IRMA sessions together. A problem can be that someone authenticates himself/herself multiple times with different attributes from the same IP address, which might make it possible to link the disclosed attributes to one person. For example, someone makes two purchases in the same online shop. The first purchase is a lady's night ticket of an event, and she discloses the attributes '18 years or older,' 'full name,' email address, and female.

The second purchase is a bracelet of the same event, and she discloses the attributes 'full name,' city, zip code, address, '18 years or older,' and email address. When someone made these purchases on the same day and from the same place with the same IP address, the shop could assume that one person disclosed all these attributes, by which the person lost privacy since the assumption is correct.

A solution for more privacy guarantees on the network layer is the use of an anonymity service, such as Tor, but these do add network latency [Ries et al., 2011]. Tor stands for The Onion Router and is the most popular Onion Routing based network, which is why we choose to implement it in the IRMA mobile app [Conrad and Shirazi, 2014]. Using Tor adds network latency, which can decrease the IRMA platform's usability. The question is then how much latency Tor adds. This study answers the question by performing measurements in multiple situations, by which 50% of the measurements happen when the app routes network traffic over the Tor network and the other 50% when it does not. In the same situations, the difference of the measurements with and without Tor shows the added Tor latencies.

The implementation of Tor in the IRMA mobile app gave some insights, such as the limitations the libraries have that we have used for it. The implementation also gave insights for different implementations that might be possible. Routing the IRMA mobile app's network traffic over the Tor network is also possible using an external app instead of implementing Tor in the app. This study compares both Tor routing solutions since each of them has different advantages over the other.

This document's remainder has the following structure: The first following section is preliminaries, which introduces some concepts and terms. Section 3 is about the related work to this study and contributions regarding other studies. Section 4 explains the research questions with their sub-questions, the sub-questions' research methods, and validation. Section 5 explains the setup, including changes to the IRMA mobile app to make the measurements possible, such as the Tor implementation. The section also explains some insights that the implementation gave. Section 6 gives the results of all the performed measurements with an analysis of these results. Section 7 compares routing of the IRMA mobile app's network traffic over the Tor network using an external app to routing the app's network traffic with Tor implemented in the app. Section 8 contains some discussion on the research and directions for future work, and the last one draws some conclusions.

2. Preliminaries

2.1. Roles

The IRMA platform has the following roles [Privacy by Design Foundation, 2020h]:

User Users can receive attributes from an issuer and store these in a mobile app. Also, users can authenticate themselves to a server by disclosing one or more of the stored attributes.

Issuer An issuer can distribute attributes to a user. For example, the issuer is a municipality, and it can issue the '18 years or older' attribute to a resident.

Verifier A verifier can authenticate users by validating the attributes that a user discloses. For example, the verifier is part of an alcohol webshop and asks the user to disclose

her '18 years or older' attribute, by which it can verify that the user is over 18 years old. If the verification is successful, the webshop can allow the user to buy alcohol.

Scheme manager A scheme manager signs and distributes public keys, issuer information, and credential types to verifiers and users. This party also decides which issuers can join its domain and what credential types they can distribute.

2.2. THE IRMA PLATFORM

The abbreviation IRMA stands for 'I Reveal My Attributes' and is a privacy-friendly authentication platform [Alpár et al., 2017]. The three software components within the IRMA platform are [Privacy by Design Foundation, 2020i]:

The IRMA mobile app The IRMA mobile app is an application that can run on Android and iOS devices. With this app, users can store attributes that they received from an issuer. Also, they can disclose attributes with the app to a verifier. Only the IRMA mobile app stores and cryptographically protects the user's attributes, so no server stores the user's attributes. A trusted issuer provides the attributes digitally signed, by which the integrity and authenticity of the attributes can be proven [Alpár et al., 2017] [Privacy by Design Foundation, 2020h]. Also, users can lock and unlock their app with his/her own chosen PIN code.

The IRMA server The IRMA server can have two roles, the issuer role and the verifier role. With the issuer role, the server can provide attributes to a user. With the verifier role, the server can verify the attributes that a user discloses, by which the server can authenticate the user. Also, the IRMA scheme the server uses determines if it uses a keyshare server.

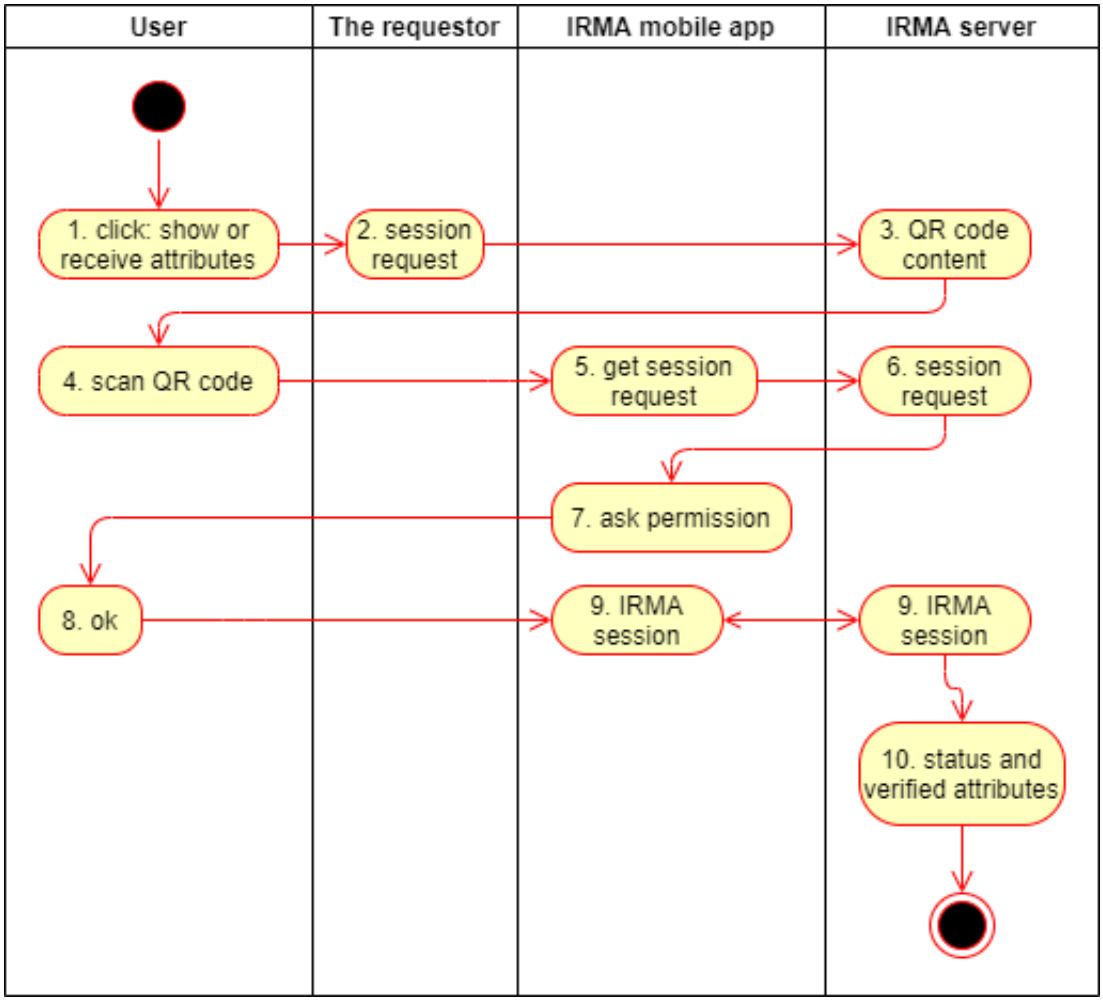
A requestor The requestor is the front-end, such as a website, to which the user wants to authenticates himself/herself. Such a front-end displays a QR code that contains information about the attributes the user needs to disclose for authentication.

Two terms used within the IRMA platform are:

Attribute-based credential Alpár et al. define attribute-based credential as "a cryptographic container of attributes that is issued and digitally signed by an issuer." [Alpár et al., 2017]. In this article, when we use the term credential, we refer to attribute-based credentials. In the current IRMA system, attribute-based credentials range from credentials with a single attribute to credentials with n attributes. For example, an email address credential only contains a single email address attribute, and a personal data credential contains attributes, such as date of birth and last name. In this example, the email credential issuer is not the issuer of the personal data credential.

IRMA scheme A container of information in the form of a directory structure, by which every party (IRMA apps, requestors, IRMA servers) within the IRMA platform can become aware of existing attribute names, credential types, and the issuers with their public keys. An IRMA scheme manager's task is to determine the information within this container and distribute it. There are currently two schemes, which are 'pbdf,'

Figure 1: The IRMA sessions flow



which is the production scheme of the Privacy by Design Foundation, and 'irma-demo,' which is the scheme for development and experimentation purposes. When we use the *irma-demo* scheme, we can also use all issuer private keys. Therefore, we can use the *irma-demo* scheme to issue attributes with our own IRMA server [Privacy by Design Foundation, 2020b].

THE IRMA SESSIONS

The IRMA platform supports three session types: disclosure sessions, attribute-based signature sessions, and issuance sessions. However, attribute-based signature sessions are out of the scope of this study. Therefore this document will not mention this session type anymore. A disclosing session has the goal of authenticating a user, and with an issuance session, the user can receive attributes on her app. The session flow of both session types are visible in Figure 1 and consists of these steps [Privacy by Design Foundation, 2020i]:

1. A common situation is that a user starts a session to disclose or receive attributes with an action on a website, such as clicking on "Log in."

2. The requestor sends a session request to the IRMA server. Part of this request is the attributes (from step 1) that the requestor wants to receive or give to the user.

3. The IRMA server accepts the session request and assigns a token to that request, and then the server returns a QR code. The QR code contains information about the URL of the IRMA server and the session type [Privacy by Design Foundation, 2020c].

4. The website displays the received QR code, which the user scans with the IRMA mobile app.

5. The IRMA mobile app requests the session request with the information from the scanned QR code [Privacy by Design Foundation, 2020e].

6. The IRMA server responds with the requested session.

7. The IRMA mobile app asks the user for permission to disclose or receive attributes.

8. The user gives permission

9. The IRMA server and the IRMA mobile app perform the IRMA protocol to receive and verify attributes or issue new attributes.

10. The IRMA server returns the session status together with the disclosed or verified attributes to the requestor.

At step 9 of the session flow, when the server verifies the attributes, a part of this task is to check that the issuer's signature that the app received together with the credential is valid, which it can do with the issuer's public key. Alternatively, when issuing at step 9, the app receives a credential with the attributes and a valid issuer signature. This signature can a verifier use combined with the issuer's public key to find out that the user has received the disclosed attributes from the issuer [Privacy by Design Foundation, 2020h].

The IRMA sessions consist of two network messages on the application layer between the IRMA mobile app and the IRMA server. One message is to start a session with an IRMA server, which will get sent after a user scans a QR code with the IRMA mobile app. When the app receives a server's response, the user can permit to disclose or receive attributes. By giving permission, the app sends the second network message that starts the IRMA protocol to disclose or receive attributes [Privacy by Design Foundation, 2020e] [Privacy by Design Foundation, 2020c].

THE KEYSHARE SERVER

In issuance and disclosure sessions, the IRMA mobile app might request the user's PIN code. A purpose of the PIN code is that a requestor can be sure that the app's owner wants to obtain or disclose attributes (and not someone else). Also, users use this same PIN code for unlocking the app. The responsibility of a keyshare server is to verify the user's PIN code and then abandon a session when the PIN is incorrect or cancel the unlocking of the app [Privacy by Design Foundation, 2020g].

Every IRMA scheme can have one IRMA keyshare server, and a manager of this scheme can decide if it uses one [Privacy by Design Foundation, 2020g].

Upon the IRMA mobile app installation, users register themselves and choose a PIN code. The app also computes a part of a secret key. A complete but masked secret key is needed for every issuance and disclosure session. Also, within the app installation, the app

performs an issuance session with the MyIRMA server from the Privacy by Design Foundation to receive the username attribute [Privacy by Design Foundation, 2020g] [Privacy by Design Foundation, 2020a].

When the IRMA keyshare server issues an attribute to the user's app for the first time, the server also creates and stores the user's second secret key part. The secret key part that the user stores and the secret key part of the user that a keyshare stores are together a full secret key. Due to the properties of the IRMA cryptography and protocol, the two halves of the secret key are never combined, yet both are always necessary for IRMA sessions. Therefore, when a user removes her secret part from a keyshare server, she cannot disclose the server's received attributes. For example, a user can log in on the MyIRMA server by providing her e-mail address on a webpage and then clicking on a received e-mail link from this server. After a user logged in on the MyIRMA server, she can remove her MyIRMA account. When a user removes her account, the user's private key part also gets removed, making it impossible to unlock the app before making a new account. Further, when attackers succeed at getting the part of the user's secret key from the user's IRMA mobile app, they do not have the full secret key. It is a protection against thieves and evil-maid attacks that users only have a part of their secret key in their app. Also, part of this protection is that users can remove their MyIRMA account [Privacy by Design Foundation, 2020g] [Privacy by Design Foundation, 2020a].

When a user closed the IRMA mobile app, the user has to unlock the app by entering her PIN upon opening the app. Also, users can lock their apps by tapping on the lock icon on the main screen, by which the app locks itself and shows a screen where users can enter their PIN to unlock the app. After the user entered her PIN, the app sends this entered PIN masked with a salt in combination with the user's user name to the keyshare server. Then the server checks if the entered PIN is correct. If the entered PIN is incorrect, the server responds that the PIN is incorrect back to the app, which the app shows this in a message to the user. If the entered PIN is correct, the server responds that the PIN is correct, together with an authentication token back to the app. When the app received the correct PIN response, the app shows its main screen, and it now has an authentication token, which is valid for 15 minutes. With this valid token, the user does not have to enter her PIN on opening the app after minimizing it, and the user does not have to use her PIN in an IRMA session. The Keyshare protocol does not explain the unlocking of the IRMA mobile app in detail, like in this paragraph. Instead, the protocol explains this unlocking as authentication to the keyshare server, which can succeed by a correctly entered PIN [Privacy by Design Foundation, 2020g] [Privacy by Design Foundation, 2020e] [Privacy by Design Foundation, 2020c].

Since users use IRMA sessions with the involvement of keyshare servers, they need to trust the keyshare servers [Privacy by Design Foundation, 2020g].

If there is a keyshare server part of the IRMA scheme, the keyshare server has a role within issuance and disclosure sessions. When the IRMA mobile app requests a session request to an IRMA server, the IRMA server responds to it with a challenge. With this challenge, the app can use it part of the secret key to compute one part of the response. The app probably has an active authentication token from unlocking the app and sends this token together with the challenge to the keyshare server. The keyshare server will then check if the token is valid. If the token is correct, the server makes a new token and calculates the second part of the response with its part of the user's key. Then the keyshare server sends

the response together with the token back to the app. Now the app can send its response back to the IRMA server, by which the response part of the app and the response part of the keyshare are part of it. The IRMA server checks if both of the received response parts are correct. If both of the response parts are correct, the session will continue. If one or both of the response parts are incorrect, the IRMA server blocks the sessions from happening [Privacy by Design Foundation, 2020g].

2.3. The anonymity service Tor

An anonymity service is a possible solution for providing more privacy guarantees on the network layer. An anonymity service works with routing network traffic from the sender through one or more nodes to the receiver. A well-known anonymity service type is the one based on Onion Routing. An Onion Routing based anonymity service works by routing network traffic through multiple nodes. The receiver of such routed network traffic can only see that traffic came from a node and not from the sender [Ries et al., 2011]. However, since network traffic gets routed through nodes, it adds network latency.

When a client wants to send a network message over an Onion Routing based network to a server, it encrypts the network message in a layer for every node. When the first two nodes in the circuit receive this message, it decrypts a layer of the message and sends the result to the next node. The third node also decrypts a layer of the message but sends the result to the server. When the server responds to the client, it sends the message to the third node in the circuit. This third node encrypts the message with a layer and sends it to the second node, and the second node also encrypts the message with a layer and sends it to the first node. Then the first node also encrypts the message with a layer and sends it to the client, by which the client can decrypt all the three layers [Conrad and Shirazi, 2014]. The first node knows only the client and the second node. The second node knows only the first node and the third node. The third node knows only the second node and the destination [Dingledine et al., 2004].

The most popular Onion Routing based network is Tor [Conrad and Shirazi, 2014]. When an application like the IRMA mobile app routes its network traffic over the Tor network, it happens through a circuit of three randomly selected nodes. Volunteers run Tor nodes, and therefore network performance of these nodes varies. Also, the distance of the client, nodes, and server among each other impacts the network latency that the use of Tor adds [Imani et al., 2017]. So, the performance latency differs for each chosen set of nodes. The first node of the circuit is the entry guard node and is the node to which a client sends a network message. The second node of the circuit is the intermediate node, and the third node of the circuit is the exit node [Conrad and Shirazi, 2014].

2.4. The HTTPS TLS protocol requires additional messages

The HTTP and HTTPS protocol's essential difference is that the HTTPS protocol uses TLS. The TLS protocol consists of two layers: the TLS record protocol and the TLS handshake protocol. The TLS record protocol encapsulates higher-level TLS protocol messages, including the TLS handshake protocol. The TLS record protocol uses data encryption to provide confidentiality and a keyed message authentication digest, by which it assures message integrity. Moreover, the TLS Record Protocol is a layer on top of a transport protocol. Therefore it influences the segments of the transport layer [Apostolopoulos et al., 1999].

In a TLS handshake between a client and a server, they send network messages be-

tween each other. The number of messages they send between each other depends on the TLS version they use. Also, the number of messages can differ if the session between client and server is a new session or session reuse. In HTTPS, the mentioned TLS Record Protocol network messages are additional messages on top of HTTP [Apostolopoulos et al., 1999] [Dierks and Allen, 1999] [Dierks and Rescorla, 2006] [Dierks and Rescorla, 2008] [Rescorla, 2018].

3. Related Work

The Tor network software changes relatively fast. For example, the Tor Project blog shows changes within the last two months for the Tor network [The Tor Project Communications Team, 2020a]. Therefore, some of the studies mentioned in this section are arguably too old. However, these studies can give an impression on the related work.

The added latency in the use of the Tor network can have a high impact on the network's usability, and there are at least two studies done within the subject Tor latency and usability. One of those studies is of Müller et al., in which they compared the performance of Internet access over the Tor network against straight Internet access, with one of the results that the Tor network is around 7.8 times slower. The measuring for this comparison happened for 28 days, with measuring latency and throughput of the Tor network by surfing to 500 popular websites from several international PlanetLab nodes executing Perl scripts. These scripts measured from almost 4.5 million requests, the HTTP requests, and the throughput of download and upload requests [Müller et al., 2012]. In a more recent study, Gallagher et al. examined the user experience of 19 students for one week with the Tor Browser (web browser that uses the Tor network) for their routine online tasks. In this examination, the students reported issues with semi-structured interviews, online questionnaires, and written self-reports. A result of this study is that all the students reported 121 user experience issues, of which eight students reported a total of 41 times that the access time was slow (latency issue) [Gallagher et al., 2018].

The anonymity network I2P (Invisible Internet Project) is also based on Onion Routing and is an alternative on Tor [Ries et al., 2011]. Both of these networks have their characteristics by which they can perform better in some aspects than others, such as network performance. Moreover, some characteristics can perform better within the network itself, and some can perform better from within the network itself to the public Internet. There are at least two studies about a comparison of Tor and I2P. The first of those two is done by Mathias Ehlert, in which he measured the bandwidth and latency of I2P and compared these results to a previous similar study by Fabian et al.. Mathias used the same Perl scripts as Fabian et al., which simulates browsing website user requests against the top 500 most visited web sites, and then saves the measured results [Fabian et al., 2010]. The study concludes that I2P can get a better latency with HTTP-GET-Requests than Tor, and Tor has a better average latency and download bandwidth than I2P [Ehlert, 2011]. The second study is from Conrad and Shirazi, which compares the aspects of both networks. The study explains that an essential difference between the services is that Tor was designed for exit traffic and that I2P tries to provide services inside its network. Therefore, I2P delivers better network performance and anonymity than Tor when communicating with users or services inside its network. Tor gives a better network performance than I2P when using the service for network traffic outside its network, such as browsing the public web [Conrad and Shirazi, 2014]. Routing the IRMA mobile app's network traffic over the Tor network

should have a better network performance than routing the app's network traffic over the I2P network.

There are anonymity services other than Tor and I2P, such as the single-hop proxies Perfect Privacy and free proxies, and another Onion Routing based network JonDonym. Ries et al. made a comparison of anonymity services. One of the things they have compared is the services' network performance, which they did with three types of measurements. They did Round Trip Time measurements with the Apache webserver benchmarking tool with measuring 200-byte requests from an HTTP server at different times in a day for six days long, and Inter-Packet Delay Variation measurements with a dedicated client-server application at every minute over four days. Also, they did a throughput measurement for three consecutive days using GNU Wget by downloading files of 100KB and 1MB. The comparison shows that the single-hop proxies have the best network performance, and Tor and I2P have the worst network performance, which is to be expected. However, the comparison also shows the usability, anonymity and security, network reliability, and economic costs. Overall in this comparison, Tor shows the best results [Ries et al., 2011].

When the added Tor network latency decreases, the usability of the network increases. A research subject is to decrease the Tor latency is Tor path selection algorithm, which is about determining the path of nodes that a client can use to build a circuit of relays. Geddes et al. did a study within this subject in which they introduce ABRA (avoiding bottleneck relay algorithm) and compare it to the congestion aware, latency aware, and vanilla online algorithms to select a circuit. They made this comparison by measuring the algorithms with Shadow, a network simulator running Tor code. They configured this network with 500 relays, 1350 web clients, 300 performance clients, 150 bulk clients, and 500 servers. The Web clients downloaded 320 KB files and paused randomly between 1 and 60000 ms. The bulk clients downloaded 5 MB files without a pause. Moreover, they divided the performance clients into three groups, and they respectively downloaded 50 KB, 1 MB, and 5 MB files, with between 1-minute pauses. Also, they used an offline algorithm that uses a model with fixed download times, which is a list of downloads that each client will download with a start and end time and an optional circuit to use for the download. With the offline algorithm, each client downloads until they reach the end time for that download. One of the conclusions of this study is that the congestion aware and latency aware algorithms perform worse than the vanilla algorithm, and ABRA performed on average between 14% and 20% better network utilization than the vanilla algorithm [Geddes et al., 2016]. The study shows interest in improving Tor, which can mean that the Tor version we use arguably can become old.

Measuring the latency that Tor adds to the use of specific software, such as the IRMA mobile app, can give insight into how much latency Tor adds to similar software. Niels Wikkerink did a study to determine that the Tor latency added to the IRMA mobile app's network traffic is acceptable. To determine the added Tor latency, he modified the app and then measured whole disclosure IRMA sessions. He started the measuring process by scanning a QR code with the IRMA mobile app. The app uses this QR code information to request a disclosure session request to the IRMA server. Since the app is modified, it starts a timer before this request. Then the server responds with the session request that the user can accept. However, Niels modified the app to accept this request automatically, and then the server performs the IRMA protocol with the app to verify the disclosed attributes. After this verification, the server sends a success message by which the app stops the timer

and saves the timed result. He did the measurements by disclosing one attribute using the Orbot app 50 times on his home network and 50 times on his university network. He used Orbot to route the network traffic of the IRMA mobile app over the Tor network. Also, he did the same 100 measurements again without using the Orbot app. The study results in that the added latency using Orbot is 0.4-0.5 seconds, which he finds acceptable since it is less than a second [Wikkerink, 2020].

We can better understand the Tor network by analyzed metrics, which the Tor Project makes publicly available through its metrics website. This website provides visualizations of analyzed statistics from the Tor network and the Tor Project infrastructure about the users, servers, traffic, performance, onion services, and applications. This website also provides services by which we can perform queries to get more information about relays or bridges in the Tor network. The Tor Project metrics team uses the Java library metrics-lib to make the analyzed statistics. The input for metrics-lib is aggregated statistics from Tor relays and bridges collected by the data collecting service CollecTor. All the collected data for the statistics follow the Tor Research Safety Board guidelines [The Tor Project Metrics Team, 2020].

3.1. CONTRIBUTIONS

The study of Niels Wikkerink is about measuring the latency that the Tor network adds to IRMA disclosure sessions. These measurements happened while the Orbot app routed the HTTP network traffic of the IRMA mobile app over the Tor network [Wikkerink, 2020]. However, the use of Tor also adds latency to the IRMA issuance sessions and the IRMA keyshare protocol. Also, in production, all IRMA traffic is routed over TLS, by which the additional handshakes required per connection could impact the latency much when routed over Tor [Apostolopoulos et al., 1999]. So, Wikkerink did not base his research setup on a realistic scenario.

My study has similarities with the study of Wikkerink. My study measures the IRMA sessions' latency, IRMA keyshare protocol, and the TLS handshake of the HTTPS protocol within the IRMA platform. All these measurements happen without the use of Orbot but with implementing the Tor protocol directly into the IRMA mobile app with the libraries go-libtor and Bine [InterPlanetary Social Network, 2020] [Retz, 2020a]. Also, my study does more and more detailed measurements than those in the study of Wikkerink. Wikkerink only measured the time of a full IRMA session so that multiple network messages end up in one measurement. My study measures the individual network messages on the application layer that are part of a full IRMA session. For each of these messages, this measuring happens by starting a timer before the request of the message and stopping the timer after receiving the response of that message.

Alpár et al. mentioned that they plan to make the IRMA mobile app communicate over Tor to protect the user's privacy at the network layer [Alpár et al., 2017]. My study shows the amount of network latency Tor adds to the app, by which they might consider an alternative to the Tor solution. My study also gives a comparison between the advantages and disadvantages of using a built-in Tor library and using an additional Tor application. We can see the project that my study delivered as a proof of concept that the IRMA mobile app can work with the Tor network using built-in libraries. Also, my study delivers some insights that the Tor implementation gives.

4. Research

We are interested in the consequences of adding Tor to IRMA. Therefore, we are going to research the following questions.

4.1. Research questions

The main research question is: How much network latency does the anonymity network Tor add within the IRMA context?

My study divided the main research question into these **sub-questions**:

- RQ1: How much network latency does TOR add to the IRMA disclosure and issuance sessions?
- RQ2: How much network latency does TOR add to the IRMA keyshare protocol of a disclosure session?
- RQ3: How much network latency does TOR add to the HTTPS protocol within the IRMA platform?
- RQ4: Which insights can we get from the effort it takes to implement Tor, within the IRMA context?
- RQ5: What are the advantages and disadvantages considering anonymity and usability of implementing Tor in the IRMA mobile app versus routing the network traffic of the app through an additional application?

RQ1: How much network latency does TOR add to the IRMA disclosure and issuance sessions?

Section 2.2 explained how the disclosure and issuance sessions work. For RQ1, these sessions' network traffic will be measured multiple times in multiple situations when the IRMA mobile app routes the HTTP network traffic over the Tor network and does not route the network traffic over Tor. However, these sessions do not get measured as a whole but as individual network messages on the application layer. Every measurement is the time it takes from that the app sends a request until it received a response from the IRMA server. Section 6 explains the measurement situations and why we choose these situations.

When a user scans a QR code from a requestor with the IRMA mobile app, the app uses this QR code information to request a disclosure or issuance session request to the IRMA server, which is the first session's network message that we measure. The user can then confirm with the app to disclose or receive the attributes, which results in the server performing the IRMA protocol with the app. Depending on the session type, this protocol verifies the disclosed attributes or the app receives attributes with it, which is the second session's network message that we measure. Section 5 explains the measuring of these two messages in more detail.

Section 2.3 explained that routing network traffic over the Tor network happens through a circuit of randomly selected nodes, by which the performance of these nodes can differ. Therefore, before every session measurement happens, the circuit will get refreshed to get an impression of the average added latency.

This study modifies the IRMA mobile app, the requestor, and the IRMA server, to make it possible to perform the automated measurements for RQ1, such as implementing the

Tor protocol in the IRMA mobile app. Section 5 explains these modifications. After the modifications, we run the IRMA server and the requestor with a configuration that includes which attributes to issue or to disclose. Section 6 mentions the attributes we used for the measurements.

The results of the measurements should give an impression of how much latency Tor can add to the IRMA sessions, which we can find in Section 6. Section 6.1 provides an analysis on these results.

RQ2: How much network latency does TOR add to the IRMA keyshare protocol of a disclosure session?

Section 2.2 mentioned that issuance and disclosing sessions can make use of the keyshare protocol, but that not all sessions make use of it. For RQ2, we want to know how much latency the keyshare protocol can add to these sessions, which we will find out by performing similar measurements as we do for RQ1. Using the keyshare protocol in IRMA sessions is optional for schemes but a default in the current production scheme. In production, it might be a choice to route the measured network messages of RQ1 over the Tor network, but not the keyshare messages. We can make this choice by using the measured latency results. So, we perform these measurements in a separate question.

The difference between RQ1 and RQ2 is that the session measurements of RQ2 measure two extra keyshare protocol network messages individually besides the RQ1 messages, which Section 5 explains in more detail. Section 6 explains more about the situations in which the measurements happen and the number of measurements in these situations.

The results of the measurements should give an impression of how much latency Tor can add to the keyshare protocol network messages of disclosure sessions, which we can find in Section 6. Section 6.1 provides an analysis on these results.

RQ3: How much network latency does TOR add to the HTTPS protocol within the IRMA platform?

This study will run the IRMA server using TLS version 1.3 for the HTTPS measurements of RQ3. As mentioned in Section 2.4, the TLS protocol of HTTPS requires additional messages on top of HTTP. For RQ3, we want to know how much latency the TLS protocol can add to the IRMA sessions for academic interests. RQ1 and RQ2 measure HTTP network traffic. This study performs for RQ3 similar measurements as we do for RQ1 and RQ2, but now over HTTPS. Section 6 explains more about the situations in which the measurements happen and the number of measurements in these situations.

The measured network latency results of RQ1 and RQ2 compared to the latency results of RQ3 can reveal the HTTP and HTTPS's latency differences, which we can do with the results in Section 6. Section 6.1 provides an analysis on these results.

In the browser Firefox, it is possible to disable the TLS 1.3 feature 0-RTT [Zimmer, 2018]. Because of the time limitations of this study's planning, we do not know if the IRMA mobile app and the IRMA server use the 0-RTT feature.

RQ4: Which insights can we get from the effort it takes to implement Tor, within the IRMA context?

To make the measurements for RQ1, RQ2, and RQ3 possible, we make some modifications to the IRMA mobile app, such as that it can route its network traffic over the Tor network. These modifications will provide insights into the challenges of incorporating Tor libraries

within the IRMA app. We can find these insights in Section 5.5. The insights are about why we choose the libraries for the modifications, with some insights about these libraries themselves, which we will find out by using them. Some other possibilities we find to implement Tor into the app are also part of these insights.

RQ5: What are the advantages and disadvantages considering anonymity and usability of implementing Tor in the IRMA mobile app versus routing the network traffic of the app through an additional application?

For this study, we implement Tor into the IRMA mobile app to make it possible to route the app's network traffic over the Tor network. However, it is also possible to route the app's network traffic over the Tor network by using an external application, such as the Orbot app [Guardian Project, 2020]. Using such an external application can provide other advantages than an implemented Tor, such as routing other apps' network traffic besides IRMA mobile over the Tor network. For example, when the user also uses the Tor network with a browser, she also gets privacy guarantees at the network layer when visiting a website. This website could be part of the same platform as the IRMA servers that the user wants to use. However, installing and configuring the external application might be too difficult for some users.

In Section 7, we discuss the nuances of this implementation decision. The section compares having Tor implemented in the IRMA mobile app to using an external Tor app.

4.2. Research method

Here follows the research method for each sub-research question.

RQ1, RQ2, RQ3 The result of these three questions depends on the average of (a big enough amount of) performed measurements. These measurements consist of keeping track of the time it takes for every answer of the server to any of the protocol messages to arrive. Since this study automatically performs the measurements, we can perform more measurements than when we would do manually, which can compensate for extreme values. Therefore, the research method for those questions is *mean quantitative research*.

RQ4 This study implements Tor in the IRMA mobile app to make Tor network measurements possible for RQ1, RQ2, and RQ3. This implementing of Tor results in some insights into the modifiability and its effort to implement it. These insights answer this question, which means that *case study* is the research method for this question.

RQ5 This question compares using a built-in Tor library and using an external Tor solution, both in combination with the IRMA mobile app. The research method for this question is *theoretical research*.

4.3. Validation

Here follows for each of the sub-research questions how the results will get validated.

RQ1, RQ2, RQ3

The questions RQ1, RQ2, and RQ3 depend on multiple measurements. Here are some actions my study took to improve the validity of these measurements:

- As mentioned in the preliminaries, the routing of network traffic over Tor happens through a circuit consisting of randomly selected nodes run by volunteers. Therefore, the network performance of these nodes differs from each other. Before every session measurement happens, the circuit will get refreshed to get an impression of the average added latency, which improves these measurements' validity.

- The IRMA mobile app will validate that when it did want to route network traffic over the Tor network, the network traffic came over the Tor network. The app can perform this validation by making an HTTP GET call to the 'Tor Check' website of the 'Tor Project' [1]. The response to this call has the information about if the app did connect to the Tor network. The app performs this call before and after it wants to route network traffic over the Tor network.

- The IRMA mobile app makes multiple measurements when it routes its network traffic over the Tor network and when it does not route over the Tor network. The more measurements, the more trustworthy the results of these measurements.

- The measurements will happen with a phone that only has a cellular or WiFi network connection active. We measure with three different WiFi connections. These four connections can vary in performance. We use these connections to check multiple realistic user scenarios since users use such connections.

- The Tor protocol is implemented directly into the IRMA mobile app via Tor libraries, instead of using an external solution such as a different app. This implementation results in not getting any possible validation problems using that external solution.

- The IRMA mobile app runs from an Android device and not from an emulator, which is a more realistic scenario.

- As already mentioned in this section, before the IRMA mobile app wants to perform a session measurement when it routes its network traffic over the Tor network, it refreshes the Tor circuit. It is a bad practice for Tor clients that the guard node changes every time Tor circuit changes since using the same guard node in every circuit change for at least 30 days in a row can improve the anonymity gained by using Tor [Elahi et al., 2012]. By monitoring the app's network traffic routed over Tor, we found that the IP address can change to which the app communicates when the Tor circuit refreshes, which means that the guard node of the circuit changes. We think that the guard node changes every time the circuit changes but that the app frequently chooses the same guard node. The changing of the guard node together with the circuit improves the results' validity.

There are some threats to the validity of the measurements needed for the three questions, which are:

- All the disclosure and issuance session measurements will happen with one IRMA server that we run. We do not use a production IRMA server for the measurements. However, multiple servers can perform the disclosure of a particular attribute. That

[1] https://check.torproject.org/

the measurements will happen with one IRMA server should not be a problem since all the servers run the same IRMA server software.

- The IRMA server and the requestor will run on the operating system (OS) Ubuntu Server, and the IRMA mobile app will run from an Android system [Canonical Ltd, 2020]. The threat is that when the IRMA software components run on a different OS or system that the performance of the different OS or system differs from those used by this study. However, this study will do measurements while the IRMA mobile app routed its network traffic over the Tor network and when it does not. Therefore, the measurements make it possible to know the Tor network's added latency, making the OS performance and the system performance, not a problem.

- This study will do all the measurements from the Netherlands. However, this study does not measure the differences in network latency between different cities and countries. For example, there might be latency differences between Maastricht and Leeuwarden, both cities in the Netherlands. So, latencies measured in Leeuwarden might match those of this study closely, but the latencies in Maastricht might be noticeable worse. We think that the chance of noticeable latency differences is small between cities in the same country. Tor Metrics can show us that France and the United States have a similar amount of nodes. Therefore, it should not be a problem that the measurements happened only from the Netherlands. The Tor Metrics website name the nodes as 'relays' [The Tor Project Metrics Team, 2020]. Also, IRMA is currently only usable within the Netherlands, so testing from the Netherlands is most realistic.

- There can be a replicability problem with the measurements that will happen when the IRMA mobile app route network traffic over Tor. Some of the Tor nodes used while the measurements happen may get taken down soon, which is realistic since volunteers run the nodes [Imani et al., 2017]. When these nodes are down, they may not get replaced with similar nodes. In this case, similar is in network performance and the place where the node will run. Tor nodes may also get added. Therefore, when someone else does the same measurements after nodes are down or added, the results can differ. This study shows the period results of when we performed the measurements, which is from 28 September 2020 to 6 December 2020.

RQ4, RQ5

A threat to RQ4 would be that with our current implementation skills that the implementation is not optimal. However, a not optimal implementation is not a problem since the focus of RQ4 are the insights that the implementation gave.

A threat to RQ5 would be that we can not find any literature on this subject.

5. Setup

This section explains the changes my study made to the IRMA mobile app, the IRMA server, and the requestor. These changes are made to make automated measuring possible, required for answering RQ1, RQ2, and RQ3. There are four types of such automated measurements, which are measurements of disclosing and issuing sessions without routing

network traffic over Tor, and measurements of disclosing and issuing sessions while routing the network traffic over Tor.

Every measurement consists of multiple measured network messages on the application layer, by which we mean that the timers are started when the application layer decides to send a message, and these timers are stopped when the response has been delivered in the application layer. My study categorizes each of those messages. Each of those categories has a text file on the IRMA mobile app, in which the app will save the measurements. After the app finished the measurements, it sends the files in an email to 'irmamobilemeasurementtests@gmail.com'. My study registered this email address by Gmail, an email service of Google [Google, 2020a]. The app measures every network message in microseconds since measuring in milliseconds results in many duplicate measurements.

As a result of all the changes, the IRMA sessions' flow changed. This new flow is in steps visible in Figure 2. An explanation of these steps follows:

1. We can now set up the measurements by first making the requestor start an IRMA session with a command-line interface (CLI) command. Part of this command are the attributes we want to disclose or to receive. This command is part of an endless while loop, so after the requestor finished this command, the loop will activate it again.

2. The requestor sends a session request to the IRMA server. Part of this request are the attributes (from step 1) that the requestor wants to receive or give to the user. (unchanged)

3. The IRMA server accepts the session request and assigns a token to that request, and then the server returns a QR code. The QR code contains information about the URL of the IRMA server and the session type. (unchanged)

4. If the measurement process has already started, then continue with step 5. The measurement process might not have started already. In this case, the user taps on a measurement type button on the wallet (main) screen of the IRMA mobile app, which results in choosing a type for the 25 measurements. The chosen measurement type should match the started session in step 1. For example, if the user started a disclosure session in step 1, the user should choose the measurement type 'disclosure session' or 'TOR disclosure session.'

5. Start the timer for measuring the 'new session' message.

6. The IRMA mobile app requests the session request. The information for this request is hard-coded in the app.

7. The IRMA server responds with the requested session. (unchanged)

8. End the timer for measuring the 'new session' message. Determine how much time it took between the start and the end of the timer.

9. The IRMA mobile app automatically accepts (without asking permission) to disclose or receive attributes.

10. Start the timer for measuring the 'respond permission' message.

11. The IRMA server and the IRMA mobile app perform the IRMA protocol to receive and verify attributes or issue new attributes. (unchanged)

12. The IRMA server returns the session status together with the disclosed or verified attributes to the requestor. (unchanged)

13. End the timer for measuring the 'respond permission' message. Determine how much time it took between the start and the end of the timer. If the status of both the 'new session' message and the 'respond permission' message was successful, then save the determined time amount that both messages took.

14. If the 25 measurements did finish, the IRMA mobile app sends an email with measurement results. Alternatively, if those measurements did not finish, continue with step 2.

15. We can stop the endless while loop from step 1.

For RQ2, when an IRMA session uses a keyshare server, my study wants to measure keyshare server messages. These measurements happen between step 9 and 10 of Figure 2 in sub-steps, which are visible in Figure 3. We know the role of the keyshare server, as mentioned in Section 2.2, and that when the session uses a keyshare server, the app will send two extra network messages [Privacy by Design Foundation, 2020c]. We named the messages after the function names of the Golang code that makes these requests, but without the camel casing. An explanation of these sub-steps follows:

9.1 Start the timer for measuring the 'get commitments' message.

9.2 The IRMA mobile app requests the get commitments request.

9.3 The keyshare server responds to the get commitments request.

9.4 End the timer for measuring the get commitment message. Determine how much time it took between the start and the end of the timer.

9.5 Start the timer for measuring the 'get proof ps' message.

9.6 The IRMA mobile app requests the get proof ps request.

9.7 The keyshare server responds to the get proof ps request.

9.8 End the timer for measuring the get proof ps message. Determine how much time it took between the start and the end of the timer.

The IRMA mobile app consists of two parts, the front end (presentation layer) part and the back end (data access layer) part. The front end part has its separate project, which is 'irmamobile.' The IRMA server, the requestor, and the IRMA mobile app's back end are part of the project 'irmago.' The irmago project named the back end of the IRMA mobile app 'client.' The two projects with the IRMA mobile app relation is visible in Figure 4. Further, we can find both projects on the GitHub platform. The irmago project consists only of Go (Golang) code. The irmamobile project consists mostly of Dart code and code from other

Figure 2: The new flow of IRMA sessions

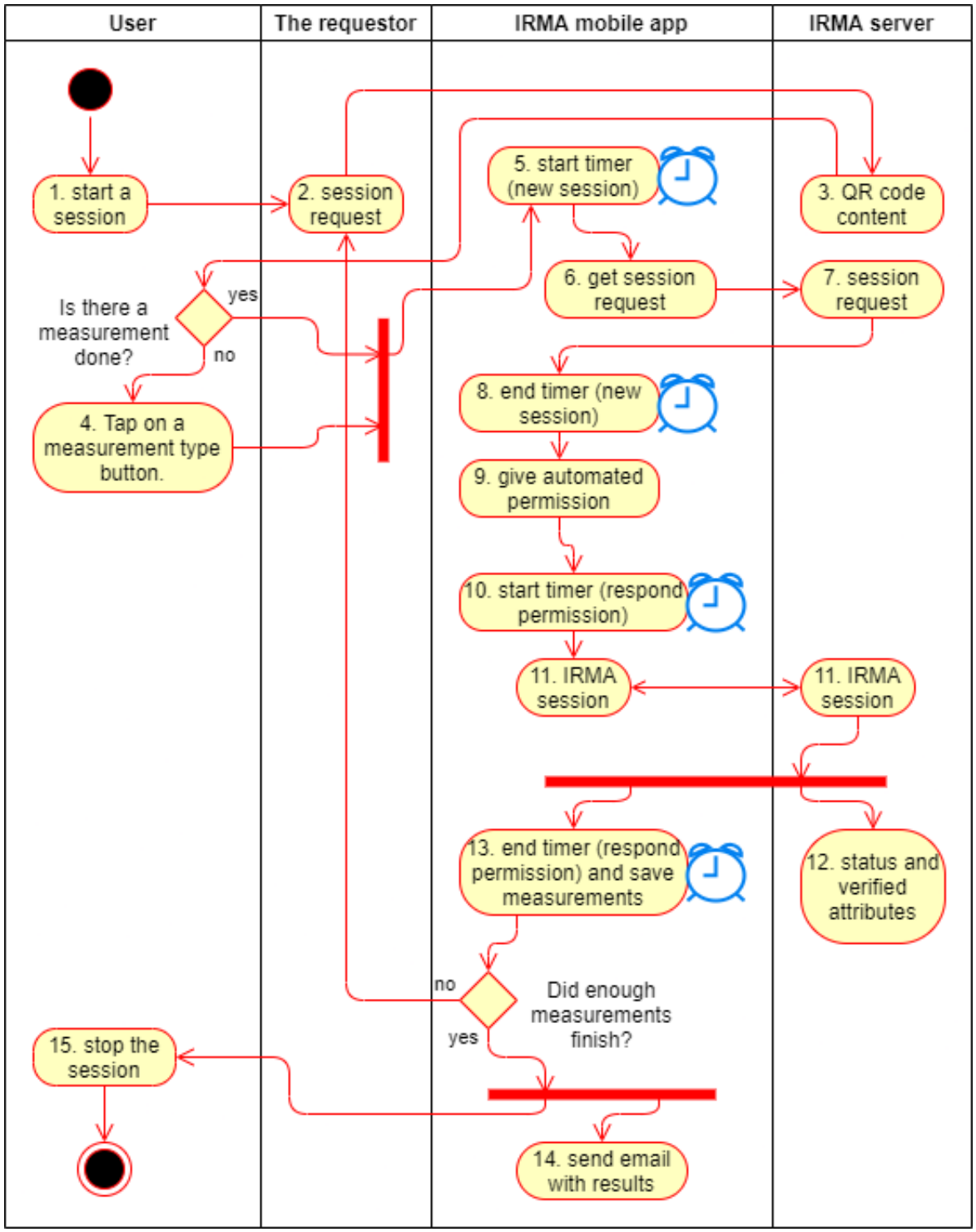


Figure 3: The keyshare server part of the new IRMA sessions flow

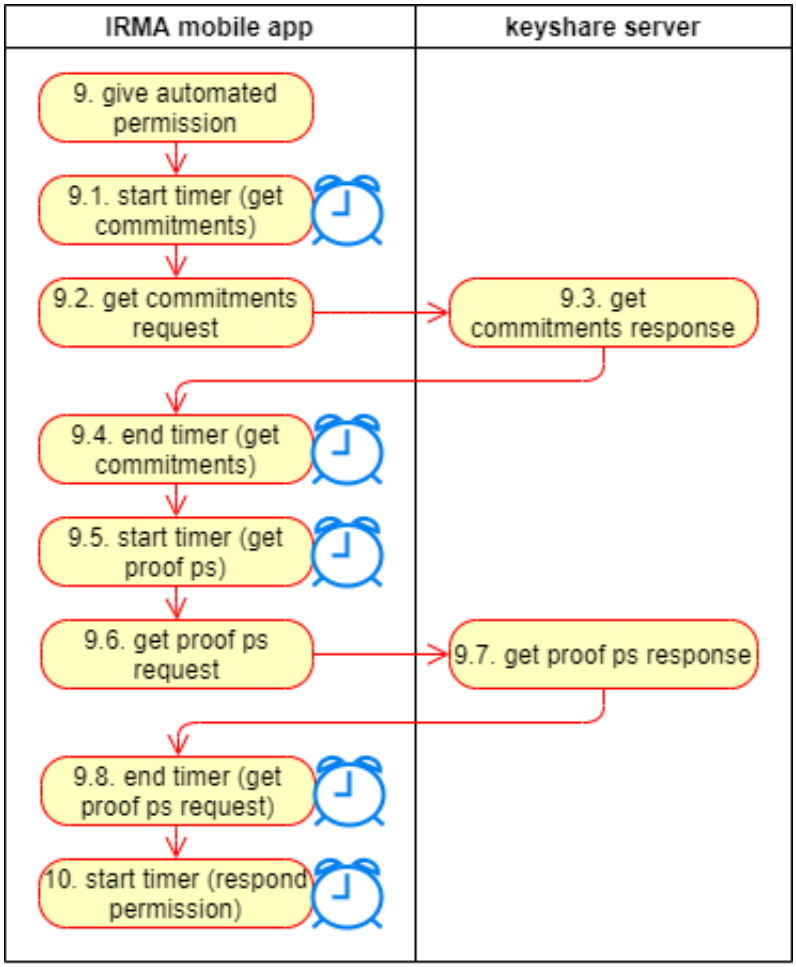
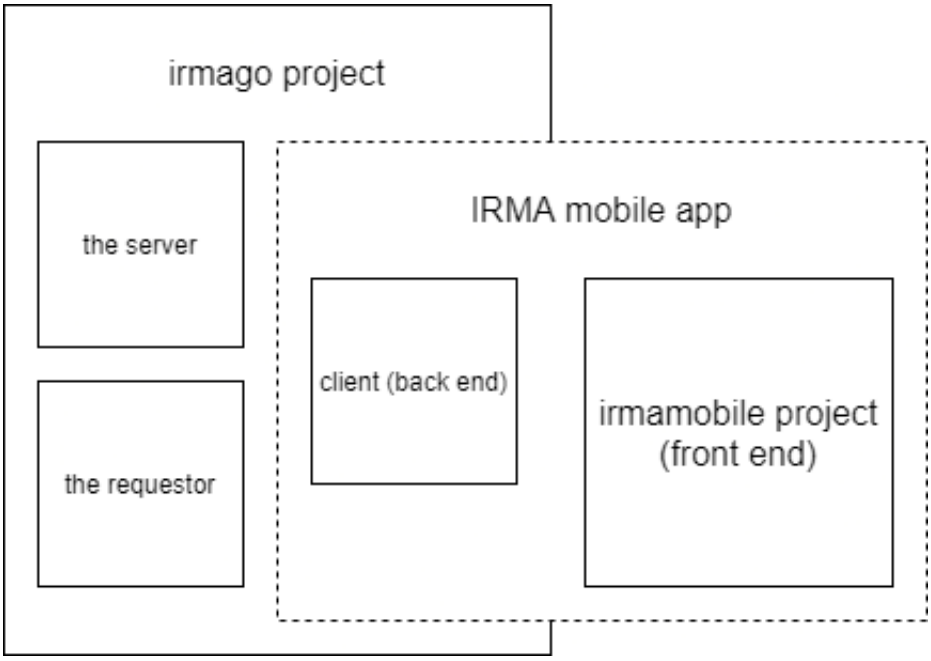


Figure 4: IRMA projects with the IRMA mobile app relation



languages, such as Go and Java [Privacy by Design Foundation, 2020e] [Privacy by Design Foundation, 2020c].

The *irmamobile* project consists mainly of two parts, the Flutter part, and the Golang part [Privacy by Design Foundation, 2020e]. Flutter is an SDK for building apps from a single codebase for Android, iOS, web, and desktop, making use of the programming language Dart [Google, 2020b]. The Golang part functions as a bridge between the Flutter part and the 'client' of the IRMA mobile app, which we can find in the project *irmago* [Privacy by Design Foundation, 2020c].

Since the development of both projects is active, my study used two specific commits of both projects. The first commit is '813d3b53b5ff409449488ac74f7e22e27d8e8ef5' of the master branch of the irmamobile project, which was committed on 10 June 2020 [Privacy by Design Foundation, 2020f]. From the master branch of the irmago project, my study uses commit 'ffa1dd898b6f7bf9e71493a4fe4a0b5eb95d2f55', which was committed on 5 June 2020 [Privacy by Design Foundation, 2020d]. My study pushed the cloned irmago project to the GitHub 'irmago-measurements' repository of user 'markuskreukniet'. My study also pushed the cloned irmamobile project to the GitHub 'irmamobile-measurements' repository of the user markuskreukniet [Kreukniet, 2020a] [Kreukniet, 2020b]. However, my study kept the cloned projects not 100% identical. There are some changes, such as the 'README.md' file.

5.1. Additional libraries and their limitations

My study used two Go libraries in the irmago project to make it possible to route the network traffic of the IRMA mobile app over the Tor network. These libraries are 'go-libtor' and 'Bine' [2] [3]. The Bine library is a Go API by which we can use and control Tor. For example,

[2] https://github.com/ipsn/go-libtor
[3] https://github.com/cretz/bine

we can start a Tor process with this API, which uses the system's 'PATH' environment variable to find Tor. The go-libtor project is an entirely statically linked, self-contained Tor Go library and consists of Go/CGO wrappers throughout the original C/C++ Tor library and its dependencies. With the API of go-libtor, we can only start an embedded Tor instance. However, with the Bine library, we can control such a Tor instance [Retz, 2020a] [InterPlanetary Social Network, 2020].

While performing RQ1, my study found some limitations in the libraries go-libtor and Bine. These limitations with their workarounds are:

- The go-libtor project uses at the moment Tor version 0.3.5.11-dev [InterPlanetary Social Network, 2020]. This version did receive the latest security fixes, but the newest stable Tor version is 0.4.3.6 [The Tor Project Communications Team, 2020b]. Therefore, this library might not be suited for the IRMA mobile app in a production environment.

- The go-libtor project or the bine project has a limitation, making it impossible to restart a Tor process. For example, it is possible to start a Tor process in an application, then shut down the application and start the application with the Tor process again. However, it is not possible to start a Tor process in an application, then kill the process in the application and start the Tor process again. Luckily, there is a workaround for this problem, which is to disable and enable the Tor network without killing the Tor process [aayushsinha44 and karalabe, 2020].

- This study wants every measurement over the Tor network on a new Tor circuit. We can determine if a circuit did change by requesting Tor's circuit information. However, we could not find a function or a method in the Bine API to get this information [Retz, 2020b]. Therefore, we made an issue in the Bine GitHub repository. The user of this repository responded to the question that it should be possible with the statement 'tor.Control.GetInfo("circuit-status")', by which 'tor' is a struct that is declared by the result of the 'tor.Start' function. As mentioned by the questioner in the issue, executing this statement results in an empty response [markuskreukniet and cretz, 2020].

  The main page of the Bine repository mentions that it is similar to Stem, a Python library by which we can control a Tor process using Tor's control protocol. Stem Docs, the Stem project website, has in their 'Tutorials' section an 'Examples' section with a 'List Circuits' example, which shows us how to get the circuits information of a Tor process [4]. In this example, we can see by looking at the naming of the 'get_circuits' method of the 'controller' object, that it should probably get the circuits information [Johnson, 2020]. In the Stem GitHub repository, we can look at the source code of the *get_circuits* method and see that the method uses the statement 'response = self.get_info('circuit-status'),' which looks similar to the statement from the Bine issue [Tor Project, 2020a].

  By debugging the 'GetInfo' method from the statement of the Bine issue, we can see that the method uses the empty 'Data' string array of the 'resp' struct to determine the

[4] https://stem.torproject.org/

result of the method. We can see also by debugging that the *resp* struct has a 'RawLines' string array, and by looking at the strings in it, they look like a 'circuit-status' result, and they look shortened and therefore unusable. Moreover, the *GetInfo* method doesn't use the *RawLines* string array. Image 5 shows the mentioned debugging of the method. With the information available from the Bine issue in combination with the debugging, my study can conclude that the *GetInfo* method does not work in its current implementation.

My study has found an incomplete workaround for not getting the circuit's information since it only gets the exit node's IP address. In this workaround, the IRMA mobile app makes an HTTP GET call to the 'ipify API,' which results in getting the application's external IP address [5]. As part of this workaround, the app also makes an HTTP GET call to the 'Tor Check' website of the 'Tor Project' [6]. The response to this call has the information about if the app did connect to the Tor network. With the two GET calls, the app can check if the Tor circuit did change.

- In torspec, which is the Tor's protocol specifications, we can see that a client can send a signal to a Tor server with the string "NEWNYM" so that Tor switches to new circuits and that new requests only use the new circuits [The Tor Project, 2020]. In the API 'package control' of the Bine library, there is a method 'Signal,' which we can use to send a signal with a string [Retz, 2020c]. My study tested this method, using the signal method to send the NEWNYM command, which results in the application "Write line: SIGNAL NEWNYM" on the console. However, the application's external IP address did not change, which means it uses the same Tor circuit as before the method call. Further, the workaround for restarting a Tor process does also work to renew circuits, which is to disable and enable the Tor network. Therefore, we chose to use this as a means of resetting the Tor circuit.

Another library my study used in the irmago project is the Go library 'email' [7]. When my study completed a batch of measurements, it wants to mail these measurements' results, by which it uses the library.

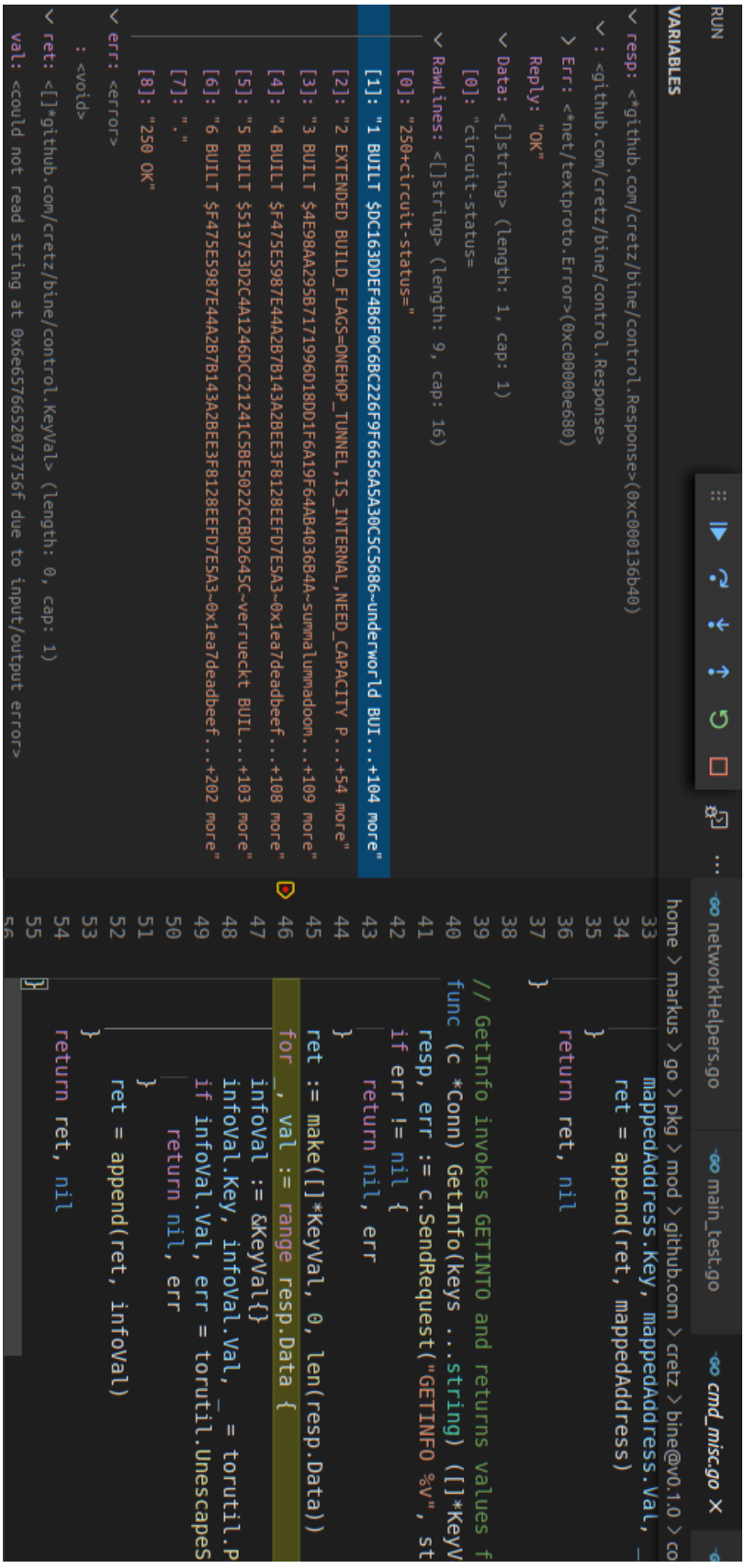
5.2. Setup for the requestor

A user can start an IRMA session with the help of a requestor. For example, when a user clicks on a button on a web page, that click can result in sending a command to start a session on an IRMA server with the requestor. Such a command can be a CLI command. The irmago project uses the Go library Cobra for such CLI commands [Francia, 2020]. At default, when the IRMA server finished a session, a user can redo the command to start a new session. However, a new session must start automatically in the same configuration as the finished one, to make automated session measurements possible.

My study uses a bash script that performs a CLI command after this same command had finished, an endless loop. When this command starts a new session, it will start a session in the same configuration again when the session ended. An explanation and some examples of commands by which we can start an IRMA session are in the IRMA docs [8].

[5] https://www.ipify.org/
[6] https://check.torproject.org/
[7] https://github.com/jordan-wright/email
[8] https://irma.app/docs/irma-cli/

Figure 5: Debugging of the *GetInfo* method



Whenever a requestor started a session with an IRMA server, the requestor has an URL by which a client can connect to the session (by scanning a QR code). At default, every time a session starts, this URL differs. The reason that this URL differs is that the IRMA server generates a random token, and this token is part of the URL. Since the automated measurements do not scan a QR code, every session that starts should have the same URL. My study modified the server code so that the token part of the session URL became a string constant, which results in the session URL becoming the same for every session.

5.3. PROJECT IRMAMOBILE MODIFICATIONS

My study modified the *irmamobile* project in the way that there is no user input needed after starting the measurements so that we can automate the measurements. The removed user input consists of scanning a QR code and tapping on buttons.

At default, when the user starts the app, the user enters her PIN code, and then the app will show its main screen, which the app names as 'WalletScreen.' On the *WalletScreen*, the user can tap on the 'QR scan' button, and the app will navigate to the 'ScannerScreen.' The *ScannerScreen* shows a QR code scanner, which the user can use to start a new issuance or disclosure session by scanning a QR code from a requestor. After scanning, the app will show the attributes the user can receive in the 'IssuanceScreen' or disclose in the 'DisclosureScreen,' which the user can accept. At successful disclosing attributes, the app shows a success message in the 'DisclosureFeedbackScreen.' By confirming this message, the app shows the *WalletScreen* again. Alternatively, at successful receiving attributes, the app shows its *WalletScreen* directly, in which the received attributes are visible [Privacy by Design Foundation, 2020e].

My study replaced the 'QR scan' button on the *WalletScreen* with nine buttons. One of these buttons kept its default behavior for the biggest part, opening the *ScannerScreen* and showing the QR code scanner. My study categorizes the other eight buttons as 'measurement buttons' (which is not visible in the app as a category). These buttons will measure multiple IRMA sessions, and each of those buttons does that in a certain situation. Such a situation is a combination of session type (disclosure or issuance), using the Tor network or not, and using an HTTP or HTTPS connection.

A measurement button opens the *ScannerScreen* without showing the QR code scanner and starts directly automated measurements with pre-configured QR code information, which is step 4 from Figure 2. After the app requested a new session from an IRMA server with the QR code information, the app automatically accepts to disclose attributes in the *DisclosureScreen* or to receive attributes in the *IssuanceScreen*, which is step 9 from Figure 2. Also, at successfully disclosing attributes, the app automatically confirms the success message in the *DisclosureFeedbackScreen* (not visible in Figure 2).

When the user tapped on the button with the default behavior, scanned a QR code, and the session finished, the app shows the *WalletScreen* again. Alternatively, when the user tapped on a measurement button and the session finished, the app will also go to the *WalletScreen* and then go directly to the *ScannerScreen*. Then the app requests a new session with the same pre-configured QR code information as the app used for the previous session. The app starts a total of 25 times a new session when the user tapped on a measurement button.

Further, tapping a measurement button will also result in that each session's network messages will get measured, as mentioned in the explanation of Figure 2. The client part

(back end) of the IRMA mobile app saves these measurements in the phone's internal storage. However, the Flutter part (front end) of the app saves these measurements in the phone's external storage since we found that external storage is easy to use with Flutter.

After the last measured session (of the 25) did finish, as already mentioned, the app sends an email. The app uses the files from internal storage as attachments in the mail. After sending the mail, the app clears its internal storage so that the results of the finished measurements are now only in the received mail and in external storage. The purpose of clearing internal storage is to reduce possible errors since the app cannot use/mail already mailed results from this place. However, the app does not clear external storage since not all app users can access the received mails.

My study found that every four minutes, the user has to authenticate herself again with her PIN code. However, in the automated measurements, not all of the 25 measurements will get finished within four minutes. Therefore, my study modified the app by which it can automatically re-authenticate the user with the PIN code '01989'. After the app re-authenticated the user, the user can continue the automated measurements by tapping on the same measurement button that started the (interrupted) measurements.

After a measured session, the app pauses itself for 13 seconds. This pause ensures that the requestor and server started a new session (steps 2 and 3 in Figure 2) before the app requests a new session (step 6 in Figure 2).

My study added the button 'WFTest' (Write File Test) to the *WalletScreen*. When we tap this button, the app writes the string 'test string' to the 'test.txt' file in external storage. We can use this button to test if the app can write to external storage before automatic measurements happen.

5.4. The IRMA mobile app client modifications

The back end of the IRMA mobile app (the client) performs the individual network messages' measurements, saves the results of these measurements, and mails the saved results after a batch of measurements is completed. My study implemented in the client three libraries with workarounds for their limitations to make the three actions possible. These libraries are 'go-libtor,' 'Bine,' and 'email,' which are all three already mentioned with their workarounds in Section 5.1.

The app saves only the measured messages of a completely successful IRMA session. So, if one of the measured messages of a session is not successful, which the project's code determines, then the app does not save any of the measured messages of that session.

Before the app wants to make any measurement over the Tor network, it checks that it has itself connected to the Tor network. If this check fails, it shuts down the whole app, and if they do not fail, the app will do the measurement. Before every measurement over the Tor network, only not before the first one, it performs the workaround to get a new Tor circuit.

5.5. Insights of the modifiability and effort of implementing Tor

The back end of the IRMA mobile app is Go code and sends network requests. Therefore, we choose a Go library, by which we could implement Tor in the back end. We could find only the Tor Go library 'go-libtor' that uses another Go library 'Bine,' which is why we choose these libraries. As already mentioned, these libraries have some limitations, which are not entirely solvable with workarounds. Some of these limitations might make the libraries not

usable for production. The library go-libtor uses an older Tor version than the current one, and that it is impossible to restart a Tor process with go-libtor.

We could not find a way to test the implemented Tor libraries locally. For example, the back end of the app has unit tests, which we could not extend with tests to measure network traffic over the Tor network. We think that a reason is that we could not test these libraries locally is that the Tor instance needs an external IP to receive a response and that an IRMA server tries to send a response to localhost over the Tor network. We tested the libraries by deploying the app on a smartphone with the libraries implemented, which takes more time than running a unit test.

Some possibilities to implement Tor in the mobile app might be:

- Implement Tor in the front end of the IRMA mobile app. The front end uses Flutter and can use plugins to extend its functionality, such as adding the Tor. For example, there is a plugin 'utopic_tor_onion_proxy' by which we can implement Tor [9].
- As already mentioned, the Bine library is similar to the Python library Stem. The Bine library has 7 contributors, 65 commits, by which the latest was on 24 January 2020 [Retz, 2020a]. The Stem library has 44 contributors, 4102 commits, by which the latest was on 8 November 2020 [Tor Project, 2020b]. Therefore, the development of Stem seems more active than Bine. It is possible to embed Python code in Go code, which might make it possible to run the Stem library from the Go code in the mobile app, and therefore a possibility to implementing Tor in the app [Chris, 2020].
- The Go library 'go-libtor' uses a wrapper consisting of Go files that include C++ sources of the 15-year-old Tor project, by which its code became more resilient over the years [Szilágyi, 2018]. Making a similar project as go-libtor or improve it is another possible Tor implementation for the app. Also, making a similar project as the Bine library or improving is a possibility.

6. Results

This study measured 1500 IRMA sessions for RQ1, RQ2, and RQ3. These sessions consist of two or four network messages on the application layer. The 1050 sessions did not use a keyshare server and consists of two messages, and 450 sessions used a keyshare server, which consists of four messages. Therefore, this study measured 3900 network messages. We can find all the results of these measurements in the 'irmamobile-measurements' repository of the user 'markuskreukniet' [10]. The 'README.md' file in the repository explains where to find the measurement results.

This study was limited to 1500 measured sessions because the automated measurements needed attention, and this amount is what the planning allowed. We measured the sessions in different situations determined by some variables, which are:

- The IRMA mobile app did or did not route the message over the Tor network. The reason for measuring with and without routing over the Tor network is to get insights into the network latency using Tor adds.

[9] https://pub.dev/packages/utopic_tor_onion_proxy
[10] https://github.com/markuskreukniet/irmamobile-measurements

- The session is a disclosure or issuance session. The reason to measure both session types is that they work differently from each other, as mentioned in Section 2.2.

- One attribute of one credential used in a session or 8 attributes that are all part of 8 different credentials used in a session. As already mentioned in Section 2.2, when the server verifies its received attributes, a part of this task is to check that the issuer's signature that the app received together with the credential is valid, which it can do with the issuer's public key. Therefore, it takes more time for the verifier to verify multiple attributes from multiple issuers than only one attribute. The network message has more bytes with multiple attributes than sending only one attribute, the added message size could result in more network latency. The reason for measuring sessions with one attribute and eight attributes is to get insights into the latency differences. The seven attribute difference between the two attribute amounts is that we choose by the assumption that disclosing one attribute is the most common scenario for most users and that seven extra attributes should give noticeable latency differences. We assumed that issuing multiple attributes would happen rarely. Therefore, this study only measured the disclosing sessions with 8 attributes.

  There are two sets of 8 attributes that my study used for the measurements, which my study named set 3 and set 4. The reason for these two sets is human error. One set was enough. The credentials part of the two sets is different, and the issuers' public keys used with verifying are also different, which might result in different network latencies.

- The phone that measured the sessions only had a cellular or WiFi network connection active. We measured with three different WiFi connections, by which the first one is a home connection, the second one is a more unstable home connection (lossy signal), and the third one is an office connection. We used one 4G connection for all the cellular network type measurements.

  WiFi can have a better download performance and lower latencies than a cellular network [Sommers and Barford, 2012]. This study did measurements with both connection types to get insights into the latency differences between the two connection types and models the IRMA app's actual usage in practice. We also wanted to check multiple realistic user scenarios and assumed that different cellular connections differ in network latency between the connections less than different WiFi connections. Therefore, we did measurements with three different WiFi connections.

- Section 2.2 mentioned that issuance and disclosing sessions can use the keyshare protocol but that not all sessions make use of it. For RQ2, we modified the IRMA mobile app to measure keyshare server (KSS) messages when a session uses a keyshare server, by which we can know how much latency the keyshare protocol can add to these sessions. These KSS messages are extra messages besides the messages of RQ1. However, we measured only disclosing sessions that used a keyshare server since we could not run a keyshare server needed for the issuance sessions.

- The session was an HTTP or HTTPS session. The reason for measuring HTTP and HTTPS sessions is academic interests and is part of the answer of RQ3.

Tor	session	attributes	network	KSS	HTTP(S)	amount
no	disclosure	1 attribute, set 1	cellular	no	HTTP	100
no	issuance	1 attribute, set 2	cellular	no	HTTP	100
yes	disclosure	1 attribute, set 1	cellular	no	HTTP	100
yes	issuance	1 attribute, set 2	cellular	no	HTTP	100
no	disclosure	1 attribute, set 1	home WiFi	no	HTTP	50
no	issuance	1 attribute, set 2	home WiFi	no	HTTP	50
yes	disclosure	1 attribute, set 1	home WiFi	no	HTTP	50
yes	issuance	1 attribute, set 2	home WiFi	no	HTTP	50
no	disclosure	8 attributes, set 3	cellular	no	HTTP	100
yes	disclosure	8 attributes, set 3	cellular	no	HTTP	100
no	issuance	1 attribute, set 2	unstable WiFi	no	HTTP	50
yes	issuance	1 attribute, set 2	unstable WiFi	no	HTTP	50
no	disclosure	1 attribute, set 1	cellular	yes	HTTP	100
yes	disclosure	1 attribute, set 1	cellular	yes	HTTP	100
no	disclosure	1 attribute, set 1	cellular	yes	HTTPS	50
yes	disclosure	1 attribute, set 1	cellular	yes	HTTPS	50
no	issuance	1 attribute, set 2	cellular	no	HTTPS	25
yes	issuance	1 attribute, set 2	cellular	no	HTTPS	25
no	disclosure	8 attributes, set 3	cellular	no	HTTPS	50
yes	disclosure	8 attributes, set 3	cellular	no	HTTPS	50
no	disclosure	8 attributes, set 4	cellular	yes	HTTPS	50
yes	disclosure	8 attributes, set 4	cellular	yes	HTTPS	50
no	disclosure	1 attribute, set 1	office WiFi	yes	HTTPS	25
yes	disclosure	1 attribute, set 1	office WiFi	yes	HTTPS	25

Table 1: number of sessions measured

The number of measured sessions in some possible situations is visible in Table 1. The attributes of the attribute sets that Table 1 mentions are visible in Table 2. By looking at the attributes in Table 2, we can see their scheme. The first text part until the dot of such an attribute is the scheme. So, all the attributes this study used are part of the scheme 'irma-demo' or the scheme 'pbdf.'

From all the measured sessions, my study determined the lowest (L), highest (H), and average (A) network latency in seconds for each combination of a situation from Table 1 and network message, which is visible in Table 3 and 4. Also, figures from 6 to 15 show these network latencies in boxplot diagrams. We made these diagrams with Microsoft Excel which is part of Microsoft Office 365 [11]. In these figures the same maximum vertical value is used of almost 14 seconds to compare the latencies between the figures easier. Such a boxplot shows the quartiles of latency measurement results. These quartiles are the first quartile (25th percentile), median value (50th percentile), third quartile (75th percentile), and the lowest and highest values, excluding any outliers. In the boxplots the possible outliers are visible as small circles.

[11] https://www.microsoft.com/nl-nl/microsoft-365/excel

attribute set	attributes
1	pbdf.pbdf.irmatube.type
2	irma-demo.MijnOverheid.ageLower.over16
3	irma-demo.MijnOverheid.ageLower.over16 irma-demo.gemeente.address.street irma-demo.ideal.ideal.iban irma-demo.surf.surfdrive.eppn irma-demo.DemoDuo.demodiploma.achieved irma-demo.coronameting.login.pseudonym irma-demo.kvk.official.legalEntity irma-demo.amsterdam.openStadBallot.ballot
4	pbdf.pbdf.irmatube.type pbdf.pbdf.ideal.bic pbdf.pbdf.linkedin.firstname pbdf.pbdf.twitter.username pbdf.pbdf.mobilenumber.mobilenumber pbdf.sidn-pbdf.email.email pbdf.gemeente.personalData.firstnames pbdf.gemeente.address.city

Table 2: attribute sets

We determine the boxplot outliers as follows. First, we determine the interquartile range (IQR), the range of the middle 50% of the values. We can calculate the IQR by subtracting the first quartile value from the third quartile value. We multiply the IQR value by 1,5. Then we determine the lower outlier cut-off with the first quartile value minus the value of IQR times 1,5 and the upper outlier cut-off with the third quartile value plus the value of IQR times 1,5. Any values below lower the outlier cut-off or any values above the upper outlier cut-off are outliers [Renze, 2021] [Weisstein, 2021].

The boxplot diagram figures use abbreviations that this study based on the explanation of Figure 2. If such an abbreviation starts with a 'T,' it refers to a measurement that happened when the app routed its network traffic over the Tor network. Otherwise, the measurement happened without using Tor. The 'D' in the abbreviation refers to a Disclosure session, while an 'I' refers to an Issuance session. Both Disclosure and Issuance measurements are subdivided into the New Session message ('NS') and the Respond Permission ('RP') message. The following list explains all abbreviations in more detail:

- DNS. Part of a **disclosure** session, the measurement of a '**new session**' message. This measurement is step 5 to 8 of the new IRMA sessions flow visible in Figure 2. In the situation where 8 attributes are part of the session, these attributes are part of set 3.
- DRP. Part of a **disclosure** session, the measurement of a '**respond permission**' message. This measurement is step 10 to 13 of the new IRMA sessions flow visible in Figure 2. In the situation where 8 attributes are part of the session, these attributes are part of set 3.

- TDNS. Part of a **disclosure** session, the measurement of a '**new session**' message, when the app routed the network traffic over the **Tor** network. This measurement is step 5 to 8 of the new IRMA sessions flow visible in Figure 2. In the situation where 8 attributes are part of the session, these attributes are part of set 3.
- TDRP. Part of a **disclosure** session, the measurement of a '**respond permission**' message, when the app routed the network traffic over the **Tor** network. This measurement is step 10 to 13 of the new IRMA sessions flow visible in Figure 2. In the situation where 8 attributes are part of the session, these attributes are part of set 3.
- DNS2. Same as DNS, but the 8 attributes of set 4 are always part of the session.
- DRP2. Same as DRP, but the 8 attributes of set 4 are always part of the session.
- TDNS2. Same as TDNS, but the 8 attributes of set 4 are always part of the session.
- TDRP2. Same as TDRP, but the 8 attributes of set 4 are always part of the session.
- INS. Part of a **issuance** session, the measurement of a '**new session**' message. This measurement is step 5 to 8 of the new IRMA sessions flow visible in Figure 2.
- IRP. Part of a **issuance** session, the measurement of a '**respond permission**' message. This measurement is step 10 to 13 of the new IRMA sessions flow visible in Figure 2.
- TINS. Part of a **issuance** session, the measurement of a '**new session**' message, when the app routed the network traffic over the **Tor** network. This measurement is step 5 to 8 of the new IRMA sessions flow visible in Figure 2.
- TIRP. Part of a **issuance** session, the measurement of a '**respond permission**' message, when the app routed the network traffic over the **Tor** network. This measurement is step 10 to 13 of the new IRMA sessions flow visible in Figure 2.
- DGC. Part of a **disclosure** session, the measurement of a '**get commitments**' message. This measurement is step 9.1 to 9.4 of the keyshare server part of the new IRMA sessions flow visible in Figure 3.
- DGPP. Part of a **disclosure** session, the measurement of a '**get proof ps**' message. This measurement is step 9.5 to 9.8 of the keyshare server part of the new IRMA sessions flow visible in Figure 3.
- TDGC. Part of a **disclosure** session, the measurement of a '**get commitments**' message, when the app routed the network traffic over the **Tor** network. This measurement is step 9.1 to 9.4 of the keyshare server part of the new IRMA sessions flow visible in Figure 3.
- TDGPP. Part of a **disclosure** session, the measurement of a '**get proof ps**' message, when the app routed the network traffic over the **Tor** network. This measurement is step 9.5 to 9.8 of the keyshare server part of the new IRMA sessions flow visible in Figure 3.

Table 5 and 6 shows the network messages' average network latencies summed together and the average latency differences of the summed network messages with and without Tor.

network message	L	A	H
one attribute (set 1 and set 2), cellular network, HTTP			
disclosure new session	0,07	0,09	0,46
disclosure respond permission	0,13	0,17	0,24
Tor disclosure new session	0,17	0,57	2,59
Tor disclosure respond permission	0,3	0,68	2,7
issuance new session	0,07	0,11	0,4
issuance respond permission	0,13	0,22	0,47
Tor issuance new session	0,2	0,55	1,98
Tor issuance respond permission	0,29	0,63	1,49
KSS get commitments	0,23	0,29	0,39
KSS get proof ps	0,24	0,3	0,42
Tor KSS get commitments	0,57	2,5	11,68
Tor KSS get proof ps	0,54	1,12	3,89
one attribute (set 1 and set 2), home WiFi network, HTTP			
disclosure new session	0,03	0,06	0,14
disclosure respond permission	0,11	0,13	0,21
Tor disclosure new session	0,18	0,49	1,31
Tor disclosure respond permission	0,31	0,65	1,74
issuance new session	0,03	0,06	0,09
issuance respond permission	0,11	0,18	0,43
Tor issuance new session	0,23	0,54	3,37
Tor issuance respond permission	0,32	0,65	3,92
eight attributes (set 3), cellular network, HTTP			
disclosure new session	0,05	0,09	0,38
disclosure respond permission	0,4	0,49	0,95
Tor disclosure new session	0,24	0,49	2,11
Tor disclosure respond permission	0,58	0,96	1,89
one attribute (set 2), unstable home WiFi network, HTTP			
issuance new session	0,03	0,06	0,32
issuance respond permission	0,09	0,2	0,47
Tor issuance new session	0,12	0,57	5,1
Tor issuance respond permission	0,21	0,51	2,23

Table 3: network message latencies over HTTP

network message	L	A	H
one attribute (set 1 and set 2), cellular network, HTTPS			
disclosure new session	0,13	0,17	0,4
disclosure respond permission	0,22	0,26	0,32
Tor disclosure new session	0,38	1,24	6,37
Tor disclosure respond permission	0,58	1,16	6,71
issuance new session	0,14	0,18	0,24
issuance respond permission	0,21	0,28	0,38
Tor issuance new session	0,42	1,52	13,78
Tor issuance respond permission	0,64	0,94	1,84
KSS get commitments	0,26	0,31	0,39
KSS get proof ps	0,27	0,32	0,44
Tor KSS get commitments	0,68	2,04	11,28
Tor KSS get proof ps	0,66	1,32	7,55
eight attributes (set 3), cellular network, HTTPS			
disclosure new session	0,11	0,16	0,52
disclosure respond permission	0,46	0,56	0,99
Tor disclosure new session	0,35	0,89	5,67
Tor disclosure respond permission	0,65	1,15	2,22
eight attributes (set 4), cellular network, HTTPS			
disclosure new session	0,1	0,15	0,38
disclosure respond permission	0,81	0,97	1,16
Tor disclosure new session	0,37	0,74	1,7
Tor disclosure respond permission	1,12	1,72	7,15
KSS get commitments	0,29	0,34	0,39
KSS get proof ps	0,27	0,31	0,43
Tor KSS get commitments	0,74	1,54	11,22
Tor KSS get proof ps	0,62	1,06	3,57
one attribute (set 1), office WiFi network, HTTPS			
disclosure new session	0,03	0,08	1,05
disclosure respond permission	0,09	0,12	0,23
Tor disclosure new session	0,47	1,61	7,28
Tor disclosure respond permission	0,68	1,37	2,92
KSS get commitments	0,11	0,16	0,45
KSS get proof ps	0,1	0,17	0,26
Tor KSS get commitments	0,76	1,94	6,03
Tor KSS get proof ps	0,7	1,59	3,16

Table 4: network message latencies over HTTPS

Figure 6: one attribute, cellular network, HTTP (disclosure sessions)

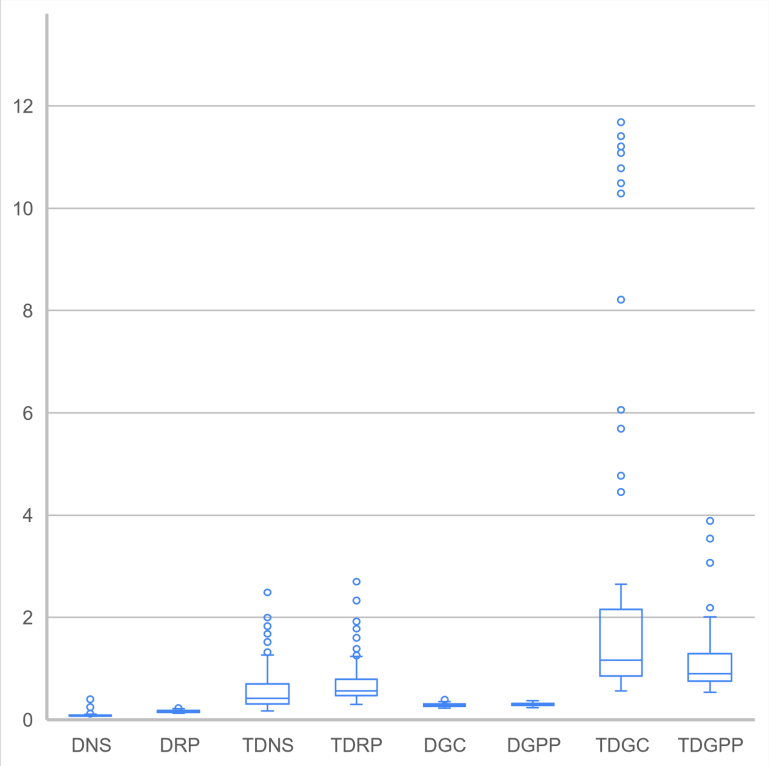


Figure 7: one attribute, cellular network, HTTP (issuance sessions)

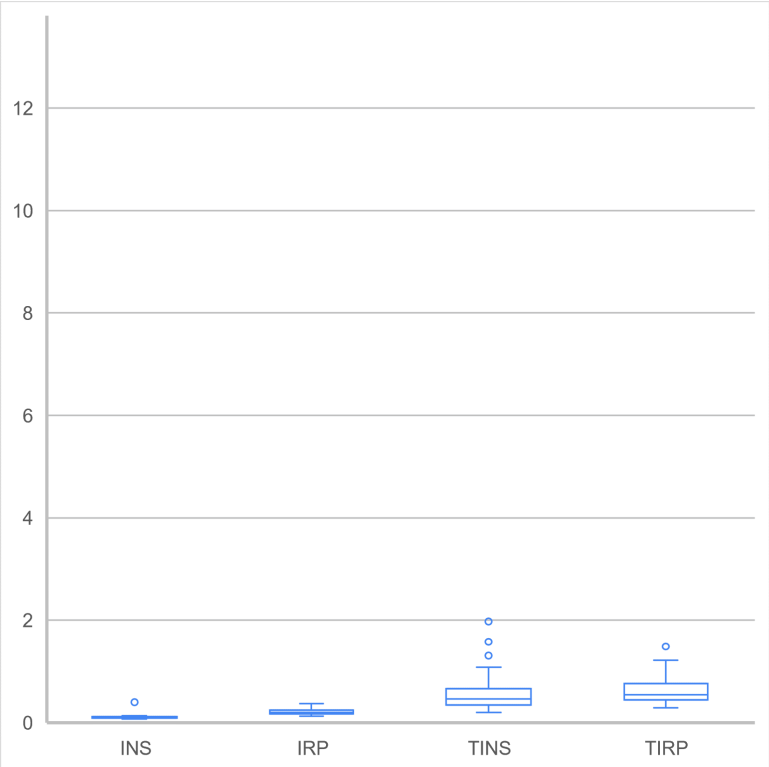


Figure 8: one attribute, home WiFi network, HTTP (disclosure and issuance sessions)

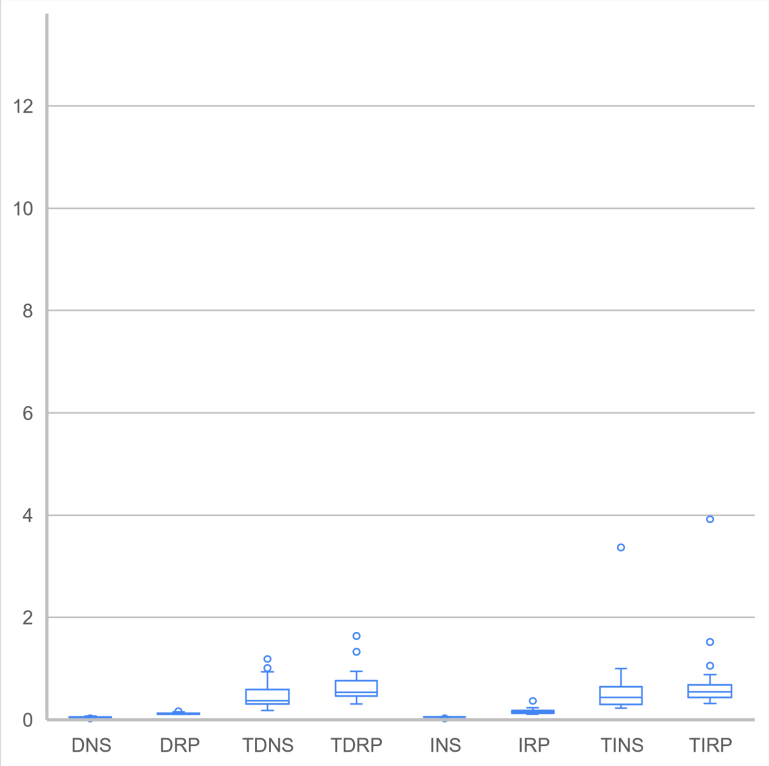


Figure 9: eight attributes (set 3), cellular network, HTTP (disclosure sessions)

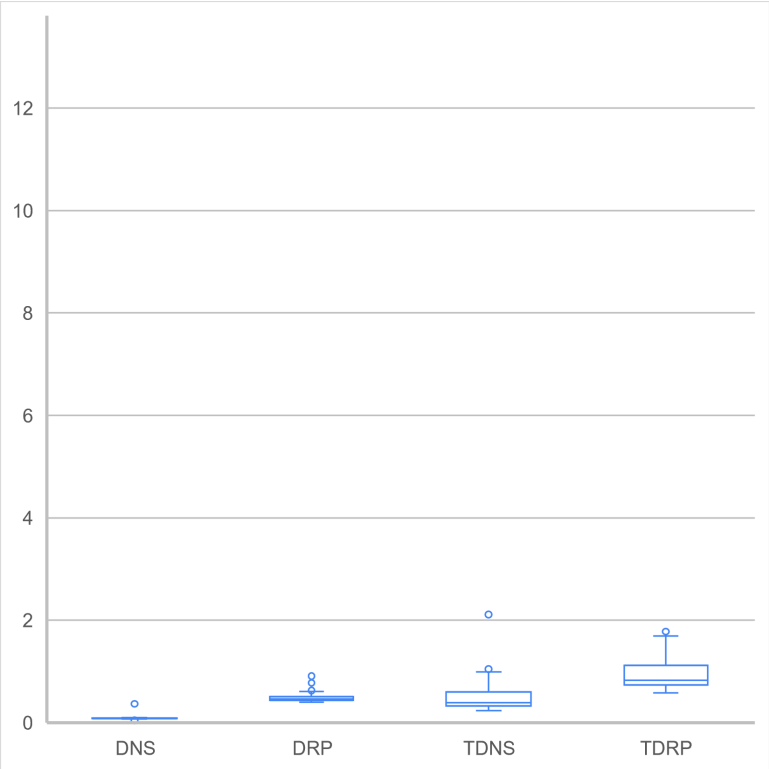


Figure 10: one attribute, unstable WiFi network, HTTP (issuance sessions)

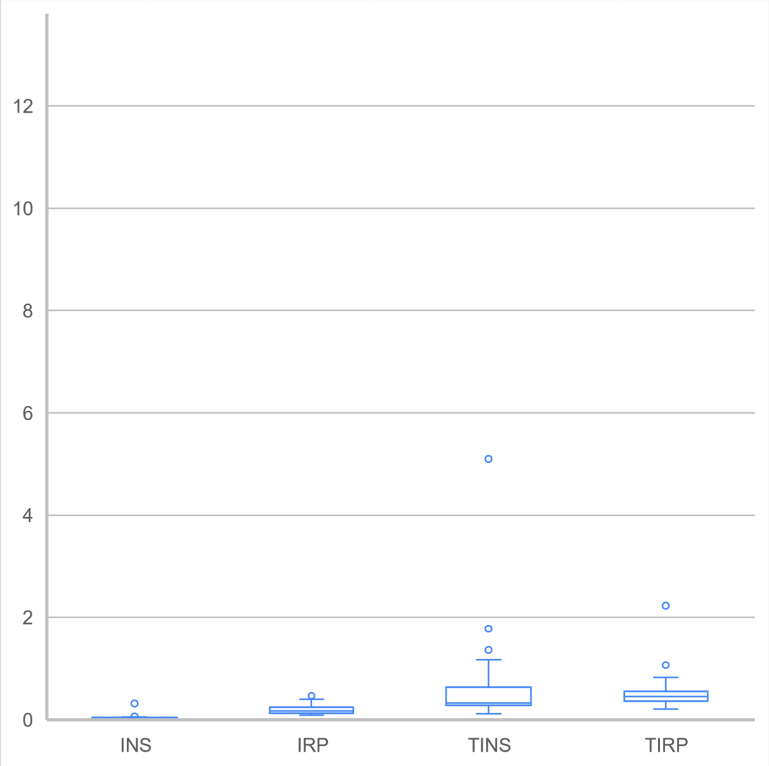


Figure 11: one attribute, cellular network, HTTPS (disclosure sessions)

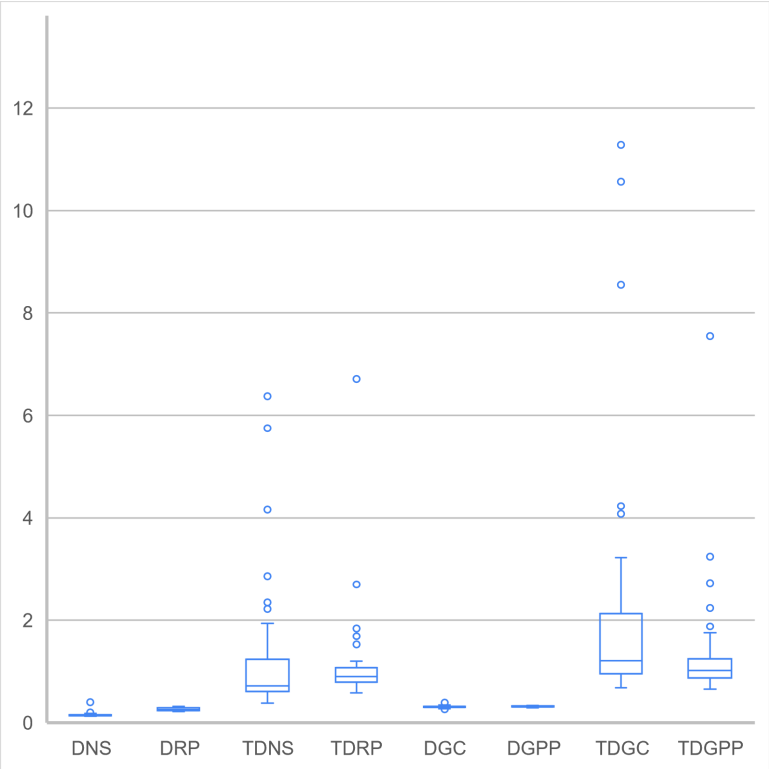


Figure 12: one attribute, cellular network, HTTPS (issuance sessions)

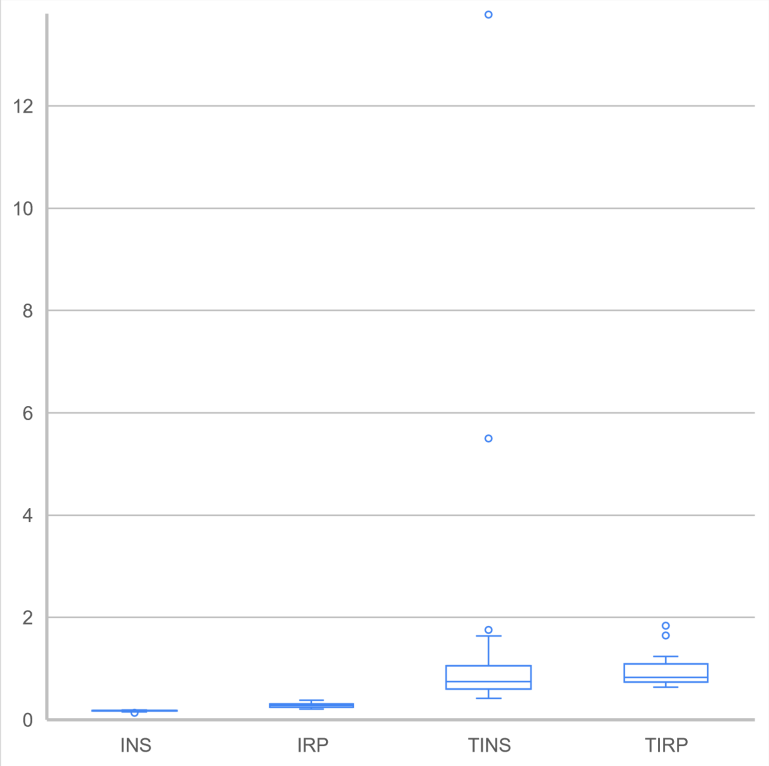


Figure 13: eight attributes (set 3), cellular network, HTTPS (disclosure sessions)

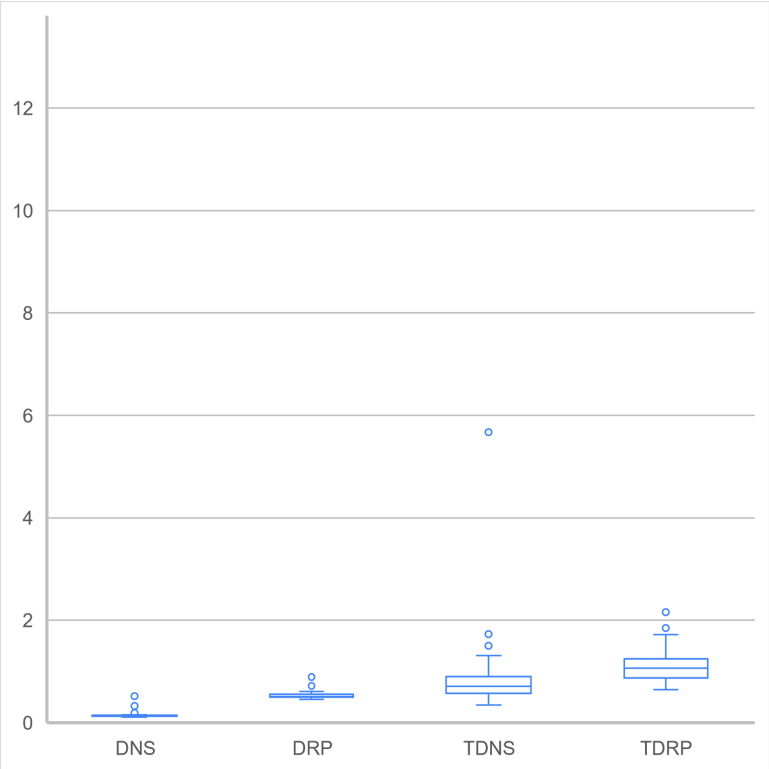


Figure 14: eight attributes (set 4), cellular network, HTTPS (disclosure sessions)

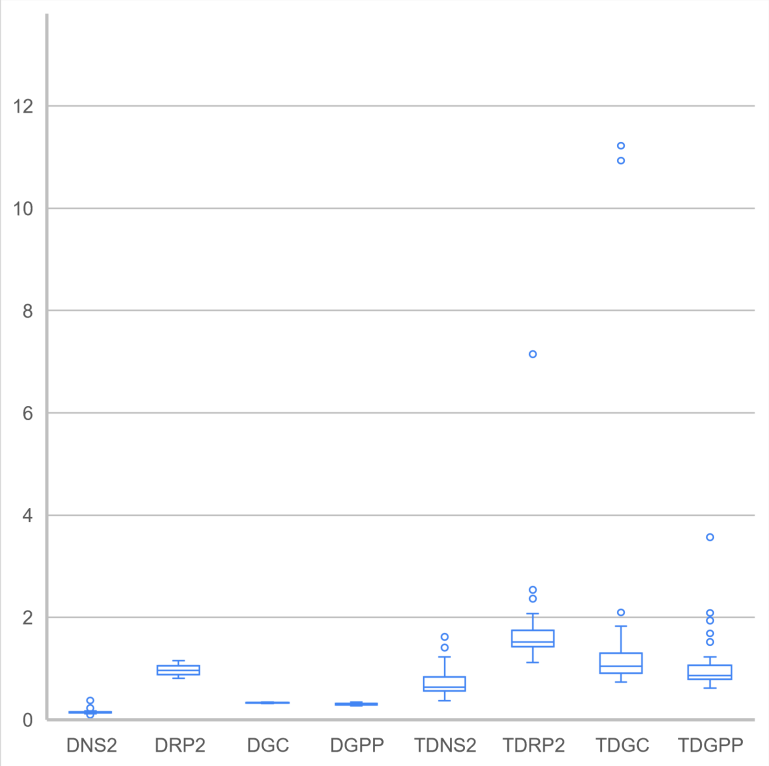
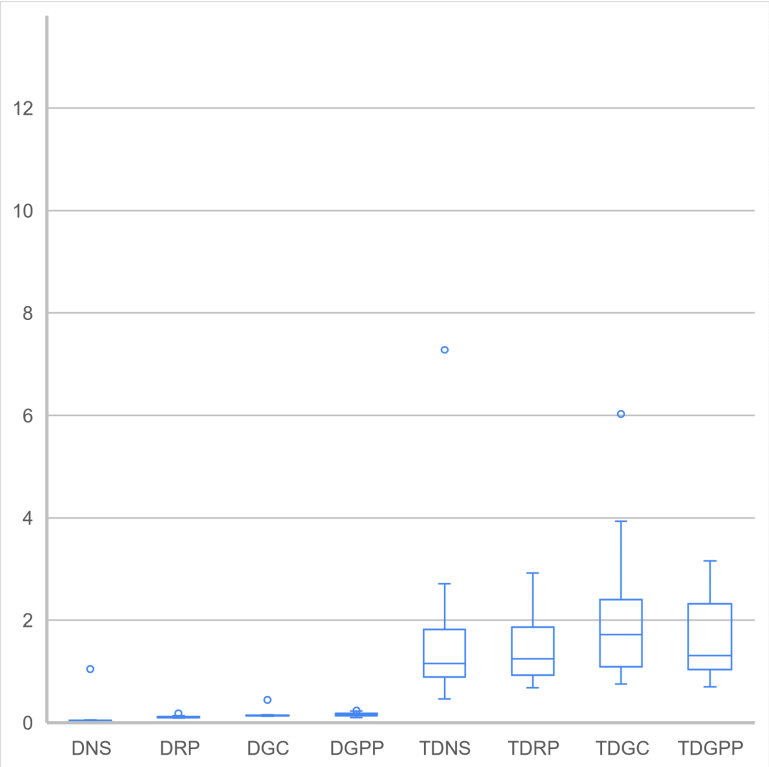


Figure 15: one attribute, office WiFi network, HTTPS (disclosure sessions)



summed network messages and added Tor latency	A
one attribute (set 1 and set 2), cellular network, HTTP	
disclosure	0,26
Tor disclosure	1,25
added Tor disclosure latency	0,99
issuance	0,33
Tor issuance	1,18
added Tor issuance latency	0,85
KSS	0,59
Tor KSS	3,62
added Tor KSS latency	3,03
one attribute (set 1 and set 2), home WiFi network, HTTP	
disclosure	0,19
Tor disclosure	1,14
added Tor disclosure latency	0,95
issuance	0,24
Tor issuance	1,19
added Tor issuance latency	0,95
eight attributes (set 3), cellular network, HTTP	
disclosure	0,58
Tor disclosure	1,45
added Tor disclosure latency	0,87
one attribute (set 2), unstable WiFi network, HTTP	
issuance	0,26
Tor issuance	1,08
added Tor issuance latency	0,82

Table 5: summed network messages over HTTP and an average of added Tor latencies

summed network messages and added Tor latency	A
one attribute (set 1 and set 2), cellular network, HTTPS	
disclosure	0,43
Tor disclosure	2,4
added Tor disclosure latency	1,97
issuance	0,46
Tor issuance	2,46
added Tor issuance latency	2
KSS	0,63
Tor KSS	3,36
added Tor KSS latency	2,73
eight attributes (set 3), cellular network, HTTPS	
disclosure	0,72
Tor disclosure	2,04
added Tor disclosure latency	1,32
eight attributes (set 4), cellular network, HTTPS	
disclosure	1,12
Tor disclosure	2,46
added Tor disclosure latency	1,34
KSS	0,65
Tor KSS	2,6
added Tor KSS latency	1,95
one attribute (set 1), office WiFi network, HTTPS	
disclosure	0,2
Tor disclosure	2,98
added Tor disclosure latency	2,78
KSS	0,33
Tor KSS	3,53
added Tor KSS latency	3,2

Table 6: summed network messages over HTTPS and an average of added Tor latencies

6.1. ANALYSIS

This section is about some observations we can make by looking at Tables 3, 4, 5, and 6.

The respond permission message's average latency is higher than the new session message's average latency. The new session message is only a request with a response, steps 6 and 7 in Figure 2. The respond permission message is more than a request with a response since it also performs the IRMA protocol, steps 9 and 11 in Figure 2. So it is completely logical that these measurements take longer, as they also measure the performance of the IRMA protocol.

The two messages of an IRMA session without a keyshare server are the new session message and the respond permission message. This study names the summed average latency of these two messages as 'AMM' and is over HTTP 0,31 seconds. AMM over HTTPS is 0,59 seconds, which means an average of extra 0,28 seconds latency for using the TLS protocol, an increase of 190,32%. AMM over HTTP and the Tor network is 1,22 seconds. AMM over HTTPS and the Tor network is 2,47 seconds, which means an average of extra 1,25 seconds latency for using TLS, an increase of 202,46%. The Tor network over HTTP adds an average latency of 0,91 seconds to AMM over HTTP without using the Tor network, which is an increase of 393,55%. Further, using the Tor network over HTTPS adds an average latency of 1,88 seconds to AMM over HTTPS without using the Tor network, which is an increase of 418,64%. So, the AMM over HTTP is lower than the AMM over HTTPS. The AMM over HTTPS is lower than the AMM over Tor and HTTP. The AMM over Tor and HTTP is lower than the AMM over Tor and HTTPS. As we expected for AMM, the added latency difference between HTTPS and HTTP is lower than using Tor and not using Tor.

The worst-case scenario of routing network traffic over the Tor network can result in latencies up to a total of 31,91 seconds, which is likely too much for a good user experience, especially without any indication that waiting so long is needed.

The lowest latencies of the respond permission messages of a disclosure session with eight attributes, with and without routing over the Tor network, are higher than the respond permission messages with one attribute in the same situation. In this case, without Tor, the latency differences between the eight attributes and one attribute vary from 0,24 seconds to 0,59 seconds. However, with Tor, the latency differences between the eight attributes and one attribute vary from 0,07 seconds to 0,54 seconds. A reason for the latency differences is the increase in message sizes, likely leading to additional IP packets being exchanged.

All the lowest latencies measured over the Tor network are higher than the average latencies of the same messages in the same situation, not over the Tor network. However, these latencies are still close together. Therefore, the lowest latencies of the messages over the Tor network can probably be unnoticeable for users.

Besides the KSS latencies, the average latency from routing over Tor of a certain message comes arguably close to the same message's highest latency without routing over Tor.

With and without routing over the Tor network, the lowest and average latencies of the home WiFi network, unstable home WiFi network, and office WiFi network are arguably almost the same.

When a disclosure session uses a keyshare server, and the app routes HTTP network traffic over the Tor network, then the Tor network will add an average of 3,03 seconds latency. However, in the same situation but over HTTPS, the Tor network will add an average of 2,63 seconds latency. So in this situation, the results show that the average latency of HTTPS is lower than HTTP, which we did not expect since the TLS record protocol of HTTPS

adds additional messages on top of HTTP, as mentioned in 2.4. This likely shows the need for more measurements, as the variability of latency added by the randomness of the Tor circuit is the likely culprit for this anomaly.

As was to be expected, the spread of latency measurements on Tor is much wider than the latency values measured without Tor. By looking at the measurement results, we can see that all the average latencies are probably not an issue for users. However, the highest latencies of these results might be too high for a good user experience, by which the sessions that used a keyshare server added the most latency. We did not expect that the network message 'Tor KSS get commitments' in the measurement results of the sessions with one attribute over the cellular network has a higher average latency over HTTP than over HTTPS. This observation is likely an indication that more measurements are needed and that the averages are very close together.

7. Implementing Tor in the app vs. an external Tor app

Implementing Tor in the IRMA mobile app has different advantages than routing the IRMA mobile app's network traffic through an external Tor app.

The main advantage of implementing Tor in the IRMA mobile app is that users do not have to install an external app and configure that application to route the IRMA mobile app network traffic through that app. When we expect a user to install and configure such an external application, she should view some instructions. For example, the IRMA mobile app can show such instructions. However, there are problems in providing such instructions to the user. For example, if the app shows these instructions, these are not visible while performing them since the app is minimized. Another example would be that the request to perform the instructions can be too much or that a user accidentally performs them wrongly.

The main advantage of routing the IRMA mobile app's network traffic through an external Tor app is that the external app can provide extra anonymity to the IRMA mobile app and other apps, such as a web browser. For example, a user visits a web page with the browser that asks the user to authenticate herself with the IRMA mobile app. The browser and the IRMA mobile app are both on the same smartphone. If only the network traffic of the IRMA mobile app routes through the external app, the web page can still track her IP address. If the network traffic of the browser and the IRMA app route through the external app, then there is no anonymity lost with the use of the browser.

There are two other possible advantages of routing the IRMA mobile app's network traffic through an external Tor app. The first advantage might be that the user understands how she can lose her anonymity. This understanding might motivate the user to use the external app and teach/motivate other people to use such an external app. The second advantage is that there is less code maintenance needed for the IRMA mobile app since the Tor project's code is not needed, but that might get a bit compensated for the code to show routing instructions.

There is also a possible addition to the implementation of Tor in the IRMA mobile app or routing the IRMA mobile app's network traffic through an external Tor app, which is to teach how users can lose privacy with a solution. For example, the IRMA mobile app explains that Tor running within the IRMA mobile app only gives extra anonymity on the network layer when using this app, not in other apps like a web browser. The solution to this problem would be to route the web browser's network traffic through an external Tor

app.

A problem with giving an explanation or instructions is that they can become outdated, which results in an extra maintenance. For example, when the IRMA mobile app has instructions about how to route the IRMA mobile app's network traffic through an external Tor app, these instructions might become outdated when that external app changes with an update.

Human error is always possible in configuring an external Tor app, resulting in the user thinking she has the extra anonymity of Tor, but she has not. Therefore, we prefer to implement Tor in the IRMA mobile app above using an external Tor app.

8. Discussion and Future work

My study did not measure the differences in network latency between different cities and countries. For example, there might be latency differences between Maastricht and Leeuwarden, both cities in the Netherlands. So, latencies measured in Leeuwarden might match those of this study closely, but the latencies in Maastricht might be noticeable worse. Also, there might be latency differences between the Netherlands and Germany. When we perform such measurements, the whole measurement situation should be the same except for the places where the measurements happen. For example, across the measurement places, we should have the same WiFi router, the same distance from the smartphone (with the IRMA app) and the router, and the same smartphone.

The most realistic measurement scenarios use a keyshare server combined with HTTPS for the advantages mentioned in Section 2, which can arguably mean that this study might not have done enough measurements in these scenarios. As mentioned in Section 6, this study missed the issuance sessions measurements that use the keyshare protocol, which is also part of these scenarios. This study implemented Tor in the IRMA mobile app with two Golang libraries, and with their limitations, they are probably not suitable for production, as mentioned in Section 5.1. A follow-up study could do more measurements in these most realistic scenarios combined with a different Tor implementation suitable for production, such as those mentioned in Section 5.5.

Routing network traffic over the Tor network adds latency, which becomes a user experience problem when Tor adds too much latency. However, we could not find any studies about how much waiting time is too much to become a user experience problem in the context of smartphone applications. Also, we could not find any studies about how to make waiting times less of a problem in smartphone applications. For example, providing a waiting screen with how long a user might have to wait could make the waiting less of a problem since they expect it.

As already mentioned in Section 7, it can be difficult for a user to route network traffic of the IRMA mobile app through an external Tor app. However, how difficult it is and how we can make it easier for users is unclear. For example, some instructions in the form of a video might be easier to follow than text and images. It is also unclear if the general smartphone/IRMA mobile app user knows she can lose anonymity using her smartphone or other devices.

This study focused only on the anonymity service Tor, but there are also other ones such as I2P and a single-hop proxy [Ries et al., 2011]. I2P has similarities with Tor [The I2P team, 2020]. However, using I2P can result in different network latencies than Tor since I2P can have lower latencies with HTTP GET requests [Ehlert, 2011]. A single-hop proxy can pro-

vide a lower network latency than Tor and I2P, but at the cost of providing less anonymity. Performing similar measurements with other anonymity services, as mentioned in Section 6 can give different results. Implementing anonymity services other than Tor gives different insights than those from Section 5.5.

9. Conclusion

For all the measured disclosure and issuance sessions with HTTP network traffic that did not use a keyshare server, Tor adds an average latency of 0,91 seconds. For all the measured sessions over HTTPS and without a keyshare server, Tor adds an average latency of 1,88 seconds. Further, for all the measured disclosure sessions with HTTP network traffic that did use a keyshare server, Tor adds an average latency of 3,03 seconds. For all the measured disclosure sessions over HTTPS and with a keyshare server, Tor adds an average latency of 2,63 seconds.

The Go libraries we have used to implement Tor into the IRMA mobile app have limitations, which are not entirely solvable with workarounds. Some of these limitations might make the libraries not usable for production. Such a limitation is that the library go-libtor uses an older Tor version than the current Tor and that it is impossible to restart a Tor process with go-libtor.

An advantage of implementing Tor in the IRMA mobile app is that users do not have to install an external app and configure that application to route the IRMA mobile app network traffic through that app. However, an advantage of routing the IRMA mobile app's network traffic through an external Tor app is that the external app can provide extra anonymity to the IRMA mobile app and other apps, such as a web browser.

This study modified the IRMA mobile app, the requestor, and the IRMA server to perform automated measurements over the Tor network and not over the Tor network. However, these automated measurements still required attention, which arguably has resulted in too few measurements in the most realistic scenarios, which are the ones that measured disclosure sessions that used a keyshare server combined with HTTPS. A take away from all the measurement results is that the latencies of the network traffic routed over the Tor network can become as high as 13,78 seconds for an individual network message, which can hurt the user experience.

The chosen libraries for the Tor implementation in the IRMA mobile app were the only available Golang libraries that we could find. This implementation and other modifications to make the automated measurements possible gave some insights, which might help with similar implementations.

This study compares implementing Tor in the IRMA mobile app and using an external Tor app, which does show the advantages of both solutions. However, we did not base this comparison on any literature and is therefore limited to our thoughts.

A future work recommendation would be to find out how much added latency by routing network traffic over the Tor network is acceptable until it becomes a user experience problem. It is unclear if the measured latencies by this study are tolerable.

The study of Niels Wikkerink is about measuring the latency that the Tor network adds to full IRMA disclosure sessions without using the keyshare protocol, which he did by routing the IRMA mobile app's HTTP network traffic over the Tor network by using an external app [Wikkerink, 2020]. This study performed similar measurements as those of Niels Wikkerink. However, this study also measured issuance sessions, the keyshare protocol (in

disclosure sessions), HTTPS network traffic, and measured the network messages individually instead of whole sessions.

Alpár et al. mentioned that they plan to make the IRMA mobile app communicate over Tor to protect the user's privacy at the network layer [Alpár et al., 2017]. My study shows the amount of network latency Tor adds to the app, delivered a comparison of an internal and external Tor solution, and delivered the internal Tor solution as a proof of concept, with some insights about its implementation.

BIBLIOGRAPHY

aayushsinha44 and karalabe. Restart tor #20. https://github.com/ipsn/go-libtor/issues/20, 2020. [Online; accessed 17-August-2020]. 21

Gergely Alpár, Fabian van den Broek, Brinda Hampiholi, Bart Jacobs, Wouter Lueks, and Sietse Ringers. Irma: practical, decentralized and privacy-friendly identity management using smartphones, 2017. 1, 3, 10, 44

George Apostolopoulos, Vinod Peris, and Debanjan Saha. Transport layer security: How much does it really cost? In *IEEE INFOCOM'99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No. 99CH36320)*, volume 2, pages 717–725. IEEE, 1999. 7, 8, 10

Canonical Ltd. Option 1: Multipass. https://ubuntu.com/download/server, 2020. [Online; accessed 28-December-2020]. 15

Chris. Embedding python in go. https://poweruser.blog/embedding-python-in-go-338c0399f3d5, 2020. [Online; accessed 21-December-2020]. 26

Bernd Conrad and Fatemeh Shirazi. A survey on tor and i2p. In *Ninth International Conference on Internet Monitoring and Protection (ICIMP2014)*, pages 22–28, 2014. 2, 7, 8

T. Dierks and C. Allen. The tls protocol version 1.0. https://www.rfc-editor.org/rfc/rfc2246.html, 1999. [Online; accessed 25-January-2020]. 8

T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.1. https://www.rfc-editor.org/rfc/rfc4346.html, 2006. [Online; accessed 25-January-2020]. 8

T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.2. https://www.rfc-editor.org/rfc/rfc5246.html, 2008. [Online; accessed 25-January-2020]. 8

Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The second-generation onion router. Technical report, Naval Research Lab Washington DC, 2004. 7

Mathias Ehlert. I2p usability vs. tor usability a bandwidth and latency comparison. In *Seminar Report, Humboldt University of Berlin*, pages 129–134, 2011. 8, 42

Tariq Elahi, Kevin Bauer, Mashael AlSabah, Roger Dingledine, and Ian Goldberg. Changing of the guards: A framework for understanding and improving entry guard selection in tor. In *Proceedings of the 2012 ACM Workshop on Privacy in the Electronic Society*, pages 43–54, 2012. 14

Benjamin Fabian, Florian Goertz, Steffen Kunz, Sebastian Müller, and Mathias Nitzsche. Privately waiting–a usability analysis of the tor anonymity network. In *SIGeBIZ track of the Americas Conference on Information Systems*, pages 63–75. Springer, 2010. 8

Steve Francia. cobra. https://github.com/spf13/cobra, 2020. [Online; accessed 14-July-2020]. 22

Kevin Gallagher, Sameer Patil, Brendan Dolan-Gavitt, Damon McCoy, and Nasir Memon. Peeling the onion's user experience layer: Examining naturalistic use of the tor browser. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 1290–1305, 2018. 8

John Geddes, Mike Schliep, and Nicholas Hopper. Abra cadabra: Magically increasing network utilization in tor by avoiding bottlenecks. In *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society*, pages 165–176, 2016. 9

Google. Get more done with gmail. https://www.google.com/intl/en/gmail/about/, 2020a. [Online; accessed 20-August-2020]. 16

Google. Technical overview. https://flutter.dev/docs/resources/technical-overview, 2020b. [Online; accessed 29-July-2020]. 20

Guardian Project. Orbot: Tor for android. https://guardianproject.info/apps/orbot/, 2020. [Online; accessed 31-March-2020]. 13

Mohsen Imani, Mehrdad Amirabadi, and Matthew Wright. The evaluation of circuit selection methods on tor. *arXiv preprint arXiv:1706.06457*, 2017. 7, 15

InterPlanetary Social Network. go-libtor - self-contained tor from go. https://github.com/ipsn/go-libtor, 2020. [Online; accessed 17-August-2020]. 10, 21

Damian Johnson. List circuits. https://stem.torproject.org/tutorials/examples/list_circuits.html, 2020. [Online; accessed 29-August-2020]. 21

Markus Kreukniet. irmago-measurements. https://github.com/markuskreukniet/irmago-measurements, 2020a. [Online; accessed 8-February-2020]. 20

Markus Kreukniet. irmamobile-measurements. https://github.com/markuskreukniet/irmamobile-measurements, 2020b. [Online; accessed 8-February-2020]. 20

markuskreukniet and cretz. How to get circuit information tor currently has available? https://github.com/cretz/bine/issues/43, 2020. [Online; accessed 14-September-2020]. 21

Sebastian Müller, Franziska Brecht, Benjamin Fabian, Steffen Kunz, and Dominik Kunze. Distributed performance measurement and usability assessment of the tor anonymization network. *Future Internet*, 4(2):488–513, 2012. 8

Privacy by Design Foundation. Myirma. https://privacybydesign.foundation/myirma/, 2020a. [Online; accessed 23-March-2020]. 6

Privacy by Design Foundation. Irma schemes. https://irma.app/docs/schemes/, 2020b. [Online; accessed 29-March-2020]. 4

Privacy by Design Foundation. irmago. https://github.com/privacybydesign/irmago, 2020c. [Online; accessed 13-July-2020]. 5, 6, 17, 20

Privacy by Design Foundation. Mark revocation tests as non-local, since they rely on an external database. https://github.com/privacybydesign/irmago/commit/ffa1dd898b6f7bf9e71493a4fe4a0b5eb95d2f55, 2020d. [Online; accessed 13-July-2020]. 20

Privacy by Design Foundation. irmamobile. https://github.com/privacybydesign/irmamobile, 2020e. [Online; accessed 28-July-2020]. 5, 6, 20, 24

Privacy by Design Foundation. Merge branch 'fix-android-back-wallet-drawn' into 'master'. https://github.com/privacybydesign/irmamobile/commit/813d3b53b5ff409449488ac74f7e22e27d8e8ef5, 2020f. [Online; accessed 13-July-2020]. 20

Privacy by Design Foundation. Keyshare protocol. https://irma.app/docs/keyshare-protocol/, 2020g. [Online; accessed 3-March-2020]. 5, 6, 7

Privacy by Design Foundation. Technical overview. https://irma.app/docs/overview/, 2020h. [Online; accessed 20-March-2020]. 2, 3, 5

Privacy by Design Foundation. What is irma? https://irma.app/docs/what-is-irma/, 2020i. [Online; accessed 18-March-2020]. 1, 3, 4

John Renze. "outlier." from mathworld–a wolfram web resource, created by eric w. weisstein. https://mathworld.wolfram.com/Outlier.html, 2021. [Online; accessed 3-March-2021]. 29

E. Rescorla. The transport layer security (tls) protocol version 1.3. https://www.rfc-editor.org/rfc/rfc8446.html, 2018. [Online; accessed 25-January-2020]. 8

Chad Retz. Bine. https://github.com/cretz/bine, 2020a. [Online; accessed 21-December-2020]. 10, 21, 26

Chad Retz. package bine. https://godoc.org/github.com/cretz/bine, 2020b. [Online; accessed 14-September-2020]. 21

Chad Retz. package control. https://godoc.org/github.com/cretz/bine/control#Conn.Signal, 2020c. [Online; accessed 2-September-2020]. 22

Thorsten Ries, Andriy Panchenko, Thomas Engel, et al. Comparison of low-latency anonymous communication systems-practical usage and performance. In *Ninth Australasian Information Security Conference*, pages 77–86. ACS, 2011. 2, 7, 8, 9, 42

Joel Sommers and Paul Barford. Cell vs. wifi: on the performance of metro area mobile connections. In *Proceedings of the 2012 Internet Measurement Conference*, pages 301–314, 2012. 27

Péter Szilágyi. A pinch of privacy: Tor from within go. https://medium.com/interplanetary-social-network/a-pinch-of-privacy-tor-from-within-go-fc4b09986120, 2018. [Online; accessed 21-December-2020]. 26

The I2P team. I2p compared to tor. https://geti2p.net/en/comparison/tor, 2020. [Online; accessed 23-April-2020]. 42

The Tor Project. Tc: A tor control protocol (version 1). https://gitweb.torproject.org/torspec.git/tree/control-spec.txt#n415, 2020. [Online; accessed 1-September-2020]. 22

The Tor Project Communications Team. Tor Blog. https://blog.torproject.org/, 2020a. [Online; accessed 20-January-2021]. 8

The Tor Project Communications Team. New releases: Tor 0.3.5.11, 0.4.2.8, and 0.4.3.6 (with security fixes). https://blog.torproject.org/new-release-tor-03511-0428-0436-security-fixes, 2020b. [Online; accessed 17-August-2020]. 21

The Tor Project Metrics Team. Tor Metrics. https://metrics.torproject.org/, 2020. [Online; accessed 21-January-2021]. 10, 15

Tor Project. torproject/stem. https://github.com/torproject/stem/blob/8634aa04114e74dc5f94875113f9b44549305368/stem/control.py#L3355-L3376, 2020a. [Online; accessed 29-August-2020]. 21

Tor Project. torproject/stem. https://github.com/torproject/stem, 2020b. [Online; accessed 21-December-2020]. 26

Eric W. Weisstein. "interquartile range." from mathworld–a wolfram web resource. https://mathworld.wolfram.com/InterquartileRange.html, 2021. [Online; accessed 3-March-2021]. 29

Niels Wikkerink. Measuring network layer anonymization on irma, 2020. Bachelor's thesis, Radboud University. 10, 43

Derek Zimmer. Supercookey – a supercookie built into tls 1.2 and 1.3. https://www.privateinternetaccess.com/blog/supercookey-a-supercookie-built-into-tls-1-2-and-1-3/, 2018. [Online; accessed 5-February-2020]. 12